Annealing Optimization for Progressive Learning With Stochastic Approximation

Christos N. Mavridis, *Member, IEEE*, and John S. Baras, *Life Fellow, IEEE*

***Abstract*—In this work, we introduce a learning model designed to meet the needs of applications in which computational resources are limited, and robustness and interpretability are prioritized. Learning problems can be formulated as constrained stochastic optimization problems, with the constraints originating mainly from model assumptions that define a tradeoff between complexity and performance. This tradeoff is closely related to overfitting, generalization capacity, and robustness to noise and adversarial attacks, and depends on both the structure and complexity of the model, as well as the properties of the optimization methods used. We develop an online prototype-based learning algorithm based on annealing optimization that is formulated as an online gradient-free stochastic approximation algorithm. The learning model can be viewed as an interpretable and progressively growing competitive-learning neural network model to be used for supervised, unsupervised, and reinforcement learning. The annealing nature of the algorithm contributes to minimal hyperparameter tuning requirements, poor local minima prevention, and robustness with respect to the initial conditions. At the same time, it provides online control over the performance–complexity tradeoff by progressively increasing the complexity of the learning model as needed, through an intuitive bifurcation phenomenon. Finally, the use of stochastic approximation enables the study of the convergence of the learning algorithm through mathematical tools from dynamical systems and control, and allows for its integration with reinforcement learning algorithms, constructing an adaptive state–action aggregation scheme.**

***Index Terms*—Optimization for machine learning, annealing optimization, online deterministic annealing (ODA), progressive learning, reinforcement learning (RL), stochastic approximation.**

I. Introduction

LEARNING from data samples has proven to be an important component in the advancement of diverse fields, including artificial intelligence, computational physics, biological sciences, communication frameworks, and cyber-physical control systems. While virtually all learning problems can be formulated as constrained stochastic optimization problems, the optimization methods can be intractable, with the constraints originating mainly from model assumptions and defining a tradeoff between complexity and performance [1]. For this reason, designing models of appropriate structure, and optimization methods with particular properties, has been the cornerstone of machine learning algorithms.

Currently, deep learning methods dominate the field of machine learning owing to their experimental performance in numerous applications [2]. However, they typically consist of overly complex models of a great many parameters, which comes at the expense of time, energy, data, memory, and computational resources [3], [4]. Furthermore, they are inherently uninterpretable and vulnerable to small perturbations [5], [6], which has led to an emerging hesitation in their usage outside common benchmark datasets and real life or security-critical applications [7]. In this work, we introduce a learning model designed to alleviate these limitations to meet the needs of applications in which computational resources are limited, and robustness and interpretability are prioritized. To that end, the learning model should create a meaningful representation, should be updated recursively (and even in real time) with easy-to-implement updates, and its complexity should be appropriately and progressively adjusted to offer online control over the tradeoff between model complexity and performance. This tradeoff is also closely related to overfitting, generalization capacity, and robustness to input perturbations and adversarial attacks [8]. This is further reinforced by recent studies revealing that existing flaws in the current benchmark datasets may have inflated the need for overly complex models [9], and that overfitting to adversarial training examples may actually hurt generalization [10].

We focus on prototype-based models [11], [12], [13], which are iterative, consistent [11], interpretable, robust [14], and topology-preserving competitive-learning neural networks [15], sparse in the sense of memory complexity, fast to train and evaluate, and have recently shown impressive robustness against adversarial attacks, suggesting suitability in security-critical applications [16]. They use a set of representatives (typically called prototypes, or codevectors) to partition the input space in an optimal way according to an appropriately defined objective function. This is an intuitive approach that parallels similar concepts from cognitive psychology and neuroscience. We approximate the global minima of the objective function by

Manuscript received 18 October 2022; accepted 11 December 2022. Date of publication 28 December 2022; date of current version 26 April 2023. This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under Grant HR00111990027, in part by ONR under Grant N00014-17-1-2622, and in part by a grant from Northrop Grumman Corporation. Recommended by Senior Editor Tetsuya Iwasaki and Guest Editors George J. Pappas, Anuradha M. Annaswamy, Manfred Morari, Claire J. Tomlin, Rene Vidal, and Melanie N. Zeilinger. *(Corresponding author: Christos N. Mavridis.)*

The authors are with the Department of Electrical and Computer Engineering and the Institute for Systems Research, University of Maryland, College Park, MD 20742 USA (e-mail: mavridis@umd.edu; baras@umd.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TAC.2022.3232706.

Digital Object Identifier 10.1109/TAC.2022.3232706

solving a sequence of optimization subproblems that make use of entropy regularization at different levels. This is a deterministic annealing (DA) approach [12], [17] that 1) adjusts the number of prototypes/neurons (which defines the complexity of the model) as needed through an intuitive bifurcation phenomenon; 2) offers robustness with respect to the initial conditions; and 3) generalizes the proximity measures used to quantify the similarity between two vectors in the data space beyond convex metrics. In addition, the annealing nature of the algorithm contributes to (but does not guarantee) avoiding poor local minima, requires minimal hyperparameter tuning, and allows online control over the performance–complexity tradeoff.

Although DA approaches have been known for a while [17], an online optimization method for such architectures is an important development, similar to the introduction of a greedy online training algorithm for a network of restricted Boltzmann machines that gave rise to one of the first effective deep learning algorithms [18]. We develop an online training rule based on a stochastic approximation algorithm [19], [20] and show that it is also gradient free, provided that the proximity measure used belongs to the family of Bregman divergences: information-theoretic dissimilarity measures that play an important role in learning applications and include the widely used Euclidean distance and Kullback–Leibler divergence [21], [22]. While stochastic approximation offers an online, adaptive, and computationally inexpensive optimization framework, it is also strongly connected to dynamical systems. This enables the study of the convergence of the learning algorithm through mathematical tools from dynamical systems and control [20]. We take advantage of this property to prove the consistency of the proposed learning algorithm as a density estimator (unsupervised learning), and as a classification rule (supervised learning). Moreover, we make use of the theory of two-timescale stochastic approximation to show that the proposed learning algorithm can be used as an adaptive aggregation scheme in reinforcement learning (RL) settings with 1) a fast component that executes a temporal-difference learning algorithm and 2) a slow component for adaptive aggregation of the state–action space. Finally, we illustrate the properties and evaluate the performance of the proposed learning algorithm in multiple experiments.

In particular, we start by dedicating Section II to reviewing the theory of stochastic approximation as an optimization approach for learning algorithms, giving emphasis to its connection to dynamical systems. This concise background is targeted toward broader audience and aims to motivate the generalization of training updates in learning algorithms. We follow with Section III, where we introduce the online deterministic annealing (ODA) algorithm for unsupervised and supervised learning and study its convergence and practical implementation. In Section IV, we show how ODA can be integrated with common RL approaches, and, in particular, as an adaptive state–action aggregation algorithm that allows Q-learning to be applied to Markov decision processes (MDPs) with infinite-dimensional state and input spaces. Section V illustrates experimental results. Finally, Section VI concludes this article.

II. Stochastic Approximation: Learning With Dynamical Systems

In this section, we briefly review the theory of stochastic approximation that is going to form the base for the convergence analysis of the proposed learning schemes. We give emphasis to its connection to dynamical systems, and how this property can be particularly useful to optimization and machine learning algorithms.

A. Stochastic Approximation and Dynamical Systems

Stochastic approximation, first introduced in [19], was originally conceived as a tool for statistical computation, and, since then, has become a central tool in a number of different disciplines, often times unbeknownst to the users, researchers, and practitioners. Stochastic approximation offers an online, adaptive, and computationally inexpensive optimization framework, properties that make it an ideal optimization method for machine learning algorithms. As a result, many of the most widely used learning algorithms partially or entirely consist of stochastic approximation algorithms, from stochastic gradient descent used in the back-propagation algorithm to train artificial neural networks [23], [24], to the Q-learning algorithm used in RL applications [25], [26]. In addition to its connection with optimization and learning algorithms, stochastic approximation is also strongly connected to dynamical systems. A fact that is often overlooked is that almost any recursive numerical algorithm can be described by a discrete time dynamical system. In this sense, results about the behavior, e.g., stability and convergence properties, of discrete time dynamical systems can be applied to iterative optimization and learning algorithms. This connection is remarkably direct in stochastic approximation that allows the study of its convergence through the analysis of an ordinary differential equation, as illustrated in the following theorem, proven in [20].

Theorem 1 ([20, Ch. 2]): Almost surely (a.s.), the sequence $\{x_n\} \in S \subseteq \mathbb{R}^d$ generated by the following stochastic approximation scheme:

$$x_{n+1} = x_n + \alpha(n)\left[h(x_n) + M_{n+1}\right],\ n \geq 0 \tag{1}$$

with prescribed x_0, *converges* to a (possibly sample-path-dependent) compact, connected, internally chain transitive, invariant set of the ODE:

$$\dot{x}(t) = h\left(x(t)\right),\ t \geq 0 \tag{2}$$

where $x : \mathbb{R}_+ \to \mathbb{R}_d$ and $x(0) = x_0$, provided the following assumptions hold.

(A1) The map $h : \mathbb{R}^d \to \mathbb{R}^d$ is Lipschitz in S, i.e., $\exists L$ with $0 < L < \infty$ such that $\|h(x) - h(y)\| \leq L\|x - y\|$, $x, y \in S$.
(A2) The stepsizes $\{\alpha(n) \in \mathbb{R}_{++},\ n \geq 0\}$ satisfy $\sum_n \alpha(n) = \infty$, and $\sum_n \alpha^2(n) < \infty$.
(A3) $\{M_n\}$ is a martingale difference sequence with respect to the increasing family of σ-fields $\mathcal{F}_n := \sigma(x_m, M_m,\ m \leq n)$,

$n \geq 0$, i.e., $\mathbb{E}[M_{n+1}|\mathcal{F}_n] = 0\, a.s.$, for all $n \geq 0$, and, furthermore, $\{M_n\}$ are square-integrable with $\mathbb{E}[\|M_{n+1}\|^2|\mathcal{F}_n] \leq K(1 + \|x_n\|^2)$, $a.s.$, where $n \geq 0$ for some $K > 0$.

(A4) The iterates $\{x_n\}$ remain bounded a.s., i.e., $\sup_n \|x_n\| < \infty\, a.s.$

Intuitively, the stochastic process (1) can be seen as a noisy discretization (also known as the Euler scheme in numerical analysis literature) of (2). As an immediate result, the following corollary also holds.

Corollary 1.1 ([20]): If the only internally chain transitive invariant sets for (2) are isolated equilibrium points, then, a.s., $\{x_n\}$ converges to a, possibly sample-dependent, equilibrium point of (2).

Given the conditions of Theorem 1 and using standard Lyapunov arguments, the following corollary, regarding distributed, asynchronous implementation of the algorithm, also holds.

Corollary 1.2 ([20, Ch. 7]): Suppose that there exists a continuously differentiable function J such that $h(x) = -\nabla J(x)$ (or $h(x) = F(x) - x$). Define $Y_n \subseteq \{1, \ldots, d\}$ to be the subset of components of x_n that are updated at time n, and $v(i, n) := \sum_{m=0}^{n} \mathbb{1}_{[i \in Y_m]}$ to be the number of times the ith component $x_n^{(i)}$ has been updated up until time n. Then, a.s., the sequence $\{x_n\}$ generated by

$$x_{n+1}^{(i)} = x_n^{(i)} + \alpha(v(i,n))\mathbb{1}_{[i \in Y_n]} \left[h^{(i)}(x_n) + M_{n+1}^{(i)}\right] \tag{3}$$

where $i \in \{1, \ldots, d\}$, and $n \geq 0$, converge to the invariant set $H := \{x : \nabla J(x) = 0\}$ (or $H := \{x : F(x) = x\}$), provided that each component (i) is updated infinitely often, i.e.,

$$\liminf_{n \to \infty} \frac{v(i,n)}{n} > 0.$$

Corollaries 1.1 and 1.2 reveal the connection of the stochastic approximation algorithms with iterative approximation and optimization algorithms, including two notable special cases: stochastic gradient descent and the Q-learning algorithm. These special cases of stochastic approximation are discussed in more detail in what follows.

B. Stochastic Gradient Descent

Stochastic gradient descent is an iterative stochastic optimization method that tries to solve the problem of minimizing the cost:

$$\min_{\theta} \mathbb{E}\left[J(X, \theta)\right] \tag{4}$$

where $X : \Omega \to H$ is a random variable defined in the probability space $(\Omega, \mathcal{F}, \mathbb{P})$, and H is a Hilbert space. The update

$$\theta_{n+1} = \theta_n - \alpha_n \nabla_\theta J(x_n, \theta_n) \tag{5}$$

is used to bypass the estimation of $\hat{\mathbb{E}}[J(x, \theta_n)] = \frac{1}{n}\sum_{i=1}^{n} J(x_i, \theta_n)$, which can be expensive or infeasible. This is a special case of a stochastic approximation update. Observe that, under the condition $\nabla_\theta \mathbb{E}[J(x, \theta_n)] = \mathbb{E}[\nabla_\theta J(x, \theta_n)]$, (5) can be written as

$$\theta_{n+1} = \theta_n + \alpha_n \left[-\nabla_\theta \mathbb{E}\left[J\right] + \left(\mathbb{E}\left[\nabla_\theta J\right] - \nabla_\theta J\right)\right] \tag{6}$$

where $h(\theta) = -\nabla_\theta \mathbb{E}[J(X, \theta)]$ is a Lipschitz continuous function, and $M_n = \mathbb{E}[\nabla_\theta J(X, \theta)] - \nabla_\theta J(x_n, \theta)$ is a martingale difference sequence, since the data samples x_n are assumed independent realizations of the random variable X. Therefore, by Theorem 1 and Corollary 1.1, as long as $\sum_n \alpha(n) = \infty$, and $\sum_n \alpha^2(n) < \infty$, and θ_n remain bounded a.s., stochastic gradient descent will converge to a possibly path-dependent equilibrium of $\dot{\theta} = -\nabla_\theta \mathbb{E}[J(X, \theta)]$, i.e., in a minimizer of $\mathbb{E}[J(X, \theta)]$.

C. Q-Learning

As a second example, the Q-learning algorithm, widely used in RL, is again a special case of a stochastic approximation algorithm [27]. Consider a discrete-time MDP $(\mathcal{X}, \mathcal{U}, \mathcal{P}, C)$ with:

- $\mathcal{X}$ being the state space;
- $\mathcal{U}$ being the action (control) space;
- $\mathcal{P} : (x, u, x') \mapsto \mathbb{P}[x'|x, u]$ being the transition probabilities associated with a stochastic state transition function $f : (x, u) \mapsto x'$;
- $C : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_+$, being the immediate cost function, assumed deterministic.

RL examines the problem of learning a control policy $u := (u_0, u_1, \ldots)$ that solves the discounted infinite-horizon optimal control problem

$$\min_u \mathbb{E}\left[\sum_{l=0}^{\infty} \gamma^l C(x_l, u_l)\right]$$

where $\gamma \in (0, 1]$. Define the value function V^u of a policy u as

$$\begin{aligned} V^u(x_k) :&= \mathbb{E}\left[\sum_{l=k}^{\infty} \gamma^{l-k} C(x_l, u_l)\right] \\ &= C(x_k, u_k) + \gamma \mathbb{E}\left[V^u(x_{k+1}) \mid x_k\right] \\ &= Q^u(x_k, u_k) \end{aligned}$$

where Q^u represents the quality function of a policy u, i.e., the expected return for taking action u_k at time k and state x_k, and thereafter following policy u. As a result of Bellman's principle, we get the (discrete-time) Hamilton–Jacobi–Bellman (HJB) equation

$$\begin{aligned} V^*(x_k) :&= \min_u \mathbb{E}\left[\sum_{l=k}^{\infty} \gamma^{l-k} C(x_l, u_l)\right] \\ &\overset{\text{(HJB)}}{=} \min_u \left\{ C(x_k, u_k) + \gamma \mathbb{E}\left[V^*(x_{k+1}) \mid x_k\right] \right\} \\ &= \min_{u_k} Q^*(x_k, u_k) \end{aligned} \tag{7}$$

where $V^* := V^{u^*}$ and $Q^* := Q^{u^*}$ represent the optimal value and Q functions, respectively. RL algorithms consist mainly of temporal-difference learning algorithms that try to approximate a solution to (7) using iterative optimization methods. The optimization is performed over a finite set of parameters that are used to describe the value (or Q) function. These parameters

typically correspond to a parametric model (e.g., a neural network) used for function approximation, or to the different values of the vector $V(\mathcal{X})$ (or $Q(\mathcal{X},\mathcal{U})$), in which case $\mathcal{X}$ and $\mathcal{U}$ are assumed finite either by definition or as a result of discretization. Assuming that the state and action spaces $\mathcal{X}$ and $\mathcal{U}$ are finite, a widely used approach is the Q-learning algorithm:

$$Q_{j+1}(x,u') = Q_j(x,u') + \alpha_j[C(x,u') + \gamma \min_u Q_j(x,' u) - Q_j(x,u')]$$

which is a stochastic approximation algorithm [27]:

$$Q_{j+1}(x,u') = Q_j(x,u') + \alpha_j \left[\left(C(x,u') + \gamma \mathbb{E}\left[\min_u Q_j(x,u)\right] - Q_j(x,u') \right) + \gamma \left(\min_u Q_j(x,' u) - \mathbb{E}\left[\min_u Q_j(x,u)\right] \right) \right] \quad (8)$$

with $h(Q(x,u')) = C(x,u') + \gamma\mathbb{E}[\min_u Q_j(x,u)] - Q_j(x,u')$, and $M_j = \gamma(\min_u Q_j(x,' u) - \mathbb{E}[\min_u Q_j(x,u)])$ is a martingale difference sequence. As a result, under the conditions of Theorem 1, the Q-learning algorithm converges to the global equilibrium of $\dot{Q} = F(Q) - Q$, with $F(Q(x,u')) = C(x,u') + \gamma\mathbb{E}[\min_u Q_j(x,u)]$, i.e., to the stationary point $Q(x,u) = C(x,u) + \gamma\mathbb{E}[\min_u Q_j(x,u)]$, which solves the HJB equation.

D. Dynamics and Control for Learning

It follows from the above that stochastic approximation algorithms define a family of iterative approximation and optimization algorithms that can be used, among others, for machine learning applications. Their strong connection to dynamical systems (see Theorem 1) can give rise to the study of learning algorithms and representations through systems-theoretic mathematics, connecting machine learning with stochastic optimization, adaptive control, and dynamical systems, which can lead to new developments in the field of machine learning. As a first example, notice that (1) defines an iterative algorithm that can be used for stochastic optimization and does not necessarily depend on the gradient of a cost function. As will be shown in Section III, this can lead to gradient-free learning algorithms that can alleviate common problems such as that of the vanishing gradients. In addition, the developed mathematical theory of dynamical systems can be utilized to construct and study learning algorithms that run at the same time but at different timescales. In particular, the theory of the ODE method for stochastic approximation in two timescales as detailed in [20] is summarized in the following theorem.

Theorem 2 ([28, Ch. 6]): Consider the sequences $\{x_n\} \in S \subseteq \mathbb{R}^d$ and $\{y_n\} \in \Sigma \subseteq \mathbb{R}^k$, generated by the iterative stochastic approximation schemes

$$x_{n+1} = x_n + \alpha(n)\left[f(x_n,y_n) + M_{n+1}^{(x)}\right] \quad (9)$$

$$y_{n+1} = y_n + \beta(n)\left[g(x_n,y_n) + M_{n+1}^{(y)}\right] \quad (10)$$

for $n \geq 0$ and $M_n^{(x)}$, $M_n^{(y)}$ martingale difference sequences, and assume that $\sum_n \alpha(n) = \sum_n \beta(n) = \infty$, $\sum_n(\alpha^2(n) + \beta^2(n)) < \infty$, and ${}^{\beta(n)}/_{\alpha(n)} \to 0$, with the last condition implying that the iterations for $\{y_n\}$ run on a slower timescale than those for $\{x_n\}$. If the equation

$$\dot{x}(t) = f(x(t),y),\ x(0) = x_0$$

has an asymptotically stable equilibrium $\lambda(y)$ for fixed y and some Lipschitz mapping λ, and the equation

$$\dot{y}(t) = g(\lambda(y(t)),y(t)),\ y(0) = y_0$$

has an asymptotically stable equilibrium y^*, then, a.s., (x_n,y_n) converges to $(\lambda(y^*),y^*)$.

This result allows for two learning algorithms, that may depend on each other, to run online at the same time, but at different timescales. As will be shown in Section IV, a two-timescale stochastic approximation algorithm can be used for RL with 1) a fast component that executes a Q-learning algorithm and 2) a slow component that adaptively partitions the state–action space according to an appropriately defined dissimilarity measure.

III. ODA for Unsupervised and Supervised Learning

To formulate the mathematics of a prototype-based learning model that progressively grows in size, it is convenient to start our analysis with the case of unsupervised learning, i.e., clustering and density estimation, and then show how these results generalize in the supervised case, i.e., in classification and regression, as well. In the context of unsupervised learning, the observations (data) are represented by a random variable $X : \Omega \to S$ defined in a probability space $(\Omega,\mathcal{F},\mathbb{P})$, where $S \subseteq \mathbb{R}^d$ is the observation space (data space). The goal of prototype-based learning is to define a similarity measure $d : S \to \text{ri}(S)$ (where $\text{ri}(S)$ represents the relative interior of S) and a set of K prototypes $\mu := \{\mu_i\}_{i=1}^K$, $\mu_i \in \text{ri}(S)$, on the data space such that an average distortion measure is minimized, i.e.,

$$\min_\mu J(\mu) := \mathbb{E}\left[\min_i d(X,\mu_i)\right]. \quad (11)$$

Here, the similarity measure as well as the number of prototypes K are predefined designer parameters. This process is equivalent to finding the most suitable model out of a set of K local constant models, and results in a piecewise-constant approximation of the data space S. This representation has been used for clustering in vector quantization applications [11], [29], and, in the limit $K \to \infty$, can be used for density estimation.

To construct a learning algorithm that progressively increases the number of prototypes K as needed according to different "levels of detail" (to be defined shortly), we will define a probability space over an infinite number of candidate models, and constrain their distribution using the maximum-entropy principle at different levels. As we will show, solving a sequence of optimization problems parameterized by a single parameter T will result in a series of model distributions with a finite number of $K(T)$ values with nonzero probability, i.e., this process results in a finite number of "effective codevectors" $K(T)$ that depends on a "temperature parameter" T.

First, we need to adopt a probabilistic approach for (11), in which a quantizer $Q : S \to \text{ri}(S)$ is defined as a discrete random

variable with domain $\mu := \{\mu_i\}_{i=1}^K$, such that (11) becomes

$$\begin{aligned}\min_{\mu} D(\mu) &:= \mathbb{E}\left[d(X,Q)\right] \\ &= \mathbb{E}\left[\mathbb{E}\left[d(X,Q)|X\right]\right] \\ &= \int p(x)\sum_i p(\mu_i|x) d_\phi(x,\mu_i)\, dx. \end{aligned} \tag{12}$$

This is now a problem of finding the locations $\{\mu_i\}$ and the association probabilities $\{p(\mu_i|x)\} := \{p(Q=\mu_i|X=x)\}$. Notice that this is a more general problem than that of (11), where it is subtly assumed that $p(\mu_i|x) = \mathbb{1}_{[x\in S_i]}$, where $S_i = \{x \in S : i = \arg\min_{j=1,\dots,K} d(x,\mu_j)\}$, $i = 1,\dots,K$, and $\{S_i\}$ defines a Voronoi partition.

Now, we make the assumption that we have access to an infinite number of possible models, i.e., that the quantizer Q is a discrete random variable over a countably infinite set $\mu := \{\mu_i\}$. Instead of choosing K a priori, we will constraint the model distribution at different levels by maximizing the entropy:

$$\begin{aligned} H(\mu) &:= \mathbb{E}\left[-\log P(X,Q)\right] \\ &= H(X) + H(Q|X) \\ &= H(X) - \int p(x)\sum_i p(\mu_i|x)\log p(\mu_i|x)\, dx. \end{aligned} \tag{13}$$

This is essentially a realization of Jaynes's maximum-entropy principle [30]. We formulate the resulting multiobjective optimization as the minimization of the Lagrangian

$$\min_{\mu} F(\mu) := D(\mu) - TH(\mu) \tag{14}$$

where $T \in [0,\infty)$ acts as a Lagrange multiplier, and, as we will show, can be seen as a temperature coefficient in an annealing process [12], [17].

Remark 1: Alternatively, (14) can be formulated as

$$\min_{\mu} F(\mu) := (1-\lambda)D(\mu) - \lambda H(\mu) \tag{15}$$

where $\lambda \in [0,1]$, with

$$T := \frac{1-\lambda}{\lambda}, \quad \lambda \in [0,1] \tag{16}$$

representing the corresponding temperature coefficient. This is a mathematically equivalent formulation that, as will be discussed, can yield major benefits in the algorithmic implementation.

Equation (14) represents the scalarization method for tradeoff analysis between two performance metrics, one related to performance and the other to generalization. The entropy H acts as a regularization term and is given progressively less weight as T decreases. For large values of $T \to \infty$, we maximize the entropy. As we will show, this results in a unique effective codevector that represents the entire data space. As T is lowered, we essentially transition from one solution of the multiobjective optimization (a Pareto point when the objectives are convex) to another in a naturally occurring direction that resembles an annealing process.

In this sense, the value of T defines the "level of detail" of the dataset that is allowed to be seen by the maximum-entropy constraint. As we will show, when certain critical values of T are reached, a bifurcation phenomenon occurs, according to which, the number of nonzero values $K(T)$ of the model distribution increases, indicating that the solution to the optimization problem of minimizing $F(T)$ requires more "effective codevectors" $K(T)$.

Remark 2: We note that this concept is similar to convex relaxation of hierarchical clustering, which results in a family of objective functions with a natural geometric interpretation [31]. However, as we will show, the proposed approach does not make any relaxation assumptions, uses entropy as a naturally occurring regularization term, and allows for the development of a gradient-free training rule based on stochastic approximation. This will result in a learning algorithm that can be integrated with RL approaches.

A. Solving the Optimization Problem

As in the case of standard vector quantization algorithms, we will minimize F by successively minimizing it first with respect to the association probabilities $\{p(\mu_i|x)\}$, and then with respect to the codevector locations μ.

The following lemma provides the solution of minimizing F with respect to the association probabilities $p(\mu_i|x)$.

Lemma 1: The solution of the optimization problem

$$\begin{aligned} F^*(\mu) &:= \min_{\{p(\mu_i|x)\}} F(\mu) \\ &\text{s.t.} \sum_i p(\mu_i|x) = 1 \end{aligned} \tag{17}$$

is given by the Gibbs distributions

$$p^*(\mu_i|x) = \frac{e^{-\frac{d(x,\mu_i)}{T}}}{\sum_j e^{-\frac{d(x,\mu_j)}{T}}} \quad \forall x \in S. \tag{18}$$

Proof: We form the Lagrangian:

$$\mathcal{L}_f(\{p(\mu_i|x)\},\nu) := D(\mu) - TH(\mu) + \nu\left(\sum_i p(\mu_i|x) - 1\right). \tag{19}$$

Taking $\frac{\partial \mathcal{L}}{\partial p(\mu|x)} = 0$ yields

$$\begin{aligned} & d(x,\mu_i) + T(1+\log p(\mu_i|x)) + \nu = 0 \\ & \Rightarrow \log p(\mu_i|x) = -\frac{1}{T}d(x,\mu_i) - \left(1+\frac{\nu}{T}\right) \\ & \Rightarrow p(\mu_i|x) = \frac{e^{-\frac{d(x,\mu_i)}{T}}}{e^{1+\frac{\nu}{T}}}. \end{aligned} \tag{20}$$

Finally, from the condition $\sum_i p(\mu_i|x) = 1$, it follows that

$$e^{1+\frac{\nu}{\lambda}} = \sum_i e^{-\frac{d(x,\mu_i)}{T}} \tag{21}$$

which completes the proof. ■

In order to minimize $F^*(\mu)$ with respect to the codevector locations μ, we set the gradients to zero

$$\begin{aligned}&\frac{d}{d\mu}F^*(\mu)=0\Rightarrow\frac{d}{d\mu}\left(D(\mu)-TH(\mu)\right)=0\\&\Rightarrow\int p(x)\sum_i\frac{d}{d\mu}\left(p^*(\mu_i|x)d_\phi(x,\mu_i)\right)\\&\qquad+T\frac{d}{d\mu}\left(p^*(\mu_i|x)\log p^*(\mu_i|x)\right)\,dx=0\\&\Rightarrow\sum_i\int p(x)p^*(\mu_i|x)\frac{d}{d\mu_i}d(x,\mu_i)\,dx=0\end{aligned}\tag{22}$$

where we have used (18), direct differentiation, and the fact that $\sum_i\frac{d}{d\mu}p^*(\mu_i|x)=\frac{d}{d\mu}\sum_i p^*(\mu_i|x)=0$. In the next section, we show that (22) has a closed form solution if the dissimilarity measure d belongs to the family of Bregman divergences.

B. Bregman Divergences as Dissimilarity Measures

Prototype-based algorithms rely on measuring the proximity between different vector representations. In most cases the Euclidean distance or another convex metric is used, but this can be generalized to alternative dissimilarity measures inspired by information theory and statistical analysis, such as the Bregman divergences [21].

Definition 1 (Bregman Divergence): Let $\phi:H\to\mathbb{R}$, be a strictly convex function defined on a vector space H such that ϕ is twice F-differentiable on H. The Bregman divergence $d_\phi:H\times H\to[0,\infty)$ is defined as

$$d_\phi(x,\mu)=\phi(x)-\phi(\mu)-\frac{\partial\phi}{\partial\mu}(\mu)(x-\mu)$$

where $x,\mu\in H$, and the continuous linear map $\frac{\partial\phi}{\partial\mu}(\mu):H\to\mathbb{R}$ is the Fréchet derivative of ϕ at μ.

In this work, we will concentrate on nonempty, compact convex sets $S\subseteq\mathbb{R}^d$ so that the derivative of d_ϕ with respect to the second argument can be written as

$$\frac{\partial d_\phi}{\partial\mu}(x,\mu)=\frac{\partial\phi(x)}{\partial\mu}-\frac{\partial\phi(\mu)}{\partial\mu}-\frac{\partial^2\phi(\mu)}{\partial\mu^2}(x-\mu)+\frac{\partial\phi(\mu)}{\partial\mu}\tag{23}$$

$$=-\frac{\partial^2\phi(\mu)}{\partial\mu^2}(x-\mu)=-\langle\nabla^2\phi(\mu),(x-\mu)\rangle\tag{24}$$

where $x,\mu\in S$, $\frac{\partial}{\partial\mu}$ represents differentiation with respect to the second argument of d_ϕ, and $\nabla^2\phi(\mu)$ represents the Hessian matrix of ϕ at μ.

Example 1: As a first example, $\phi(x)=\langle x,x\rangle,\ \ x\in\mathbb{R}^d$, yields the squared Euclidean distance $d_\phi(x,\mu)=\|x-\mu\|^2$.

Example 2: A second interesting Bregman divergence that shows the connection to information theory is the generalized I-divergence that results from $\phi(x)=\langle x,\log x\rangle$, $x\in\mathbb{R}^d_{++}$ such that

$$d_\phi(x,y)=\langle x,\log x-\log\mu\rangle-\langle\mathbb{1},x-\mu,\rangle$$

where $\mathbb{1}\in\mathbb{R}^d$ is the vector of ones. It is easy to see that $d_\phi(x)$ reduces to the Kullback–Leibler divergence if $\langle\mathbb{1},x\rangle=1$.

The family of Bregman divergences provides proximity measures that have been shown to enhance the performance of a learning algorithm [22]. There is also a deeper connection of Bregman divergences to prototype-based learning algorithms [21]. In the next theorem, we show that we can have analytical solution to the last optimization step (22) in a convenient centroid form, if d is a Bregman divergence.

Theorem 3: The optimization problem

$$\min_\mu F^*(\mu)\tag{25}$$

where $F^*(\mu)$ is defined in (17) is solved by the codevector locations μ given by

$$\mu_i^*=\mathbb{E}\left[X|\mu_i\right]=\frac{\int xp(x)p^*(\mu_i|x)\,dx}{p^*(\mu_i)}\tag{26}$$

if $d:=d_\phi$ is a Bregman divergence for some function ϕ that satisfies Definition 1.

Proof: Given (24), (22) becomes

$$\int(x-\mu_i)p(x)p^*(\mu_i|x)\,dx=0\tag{27}$$

which is equivalent to (26) since $\int p(x)p^*(\mu_i|x)\,dx=p^*(\mu_i)$.■

C. Bifurcation and the Number of Clusters

So far, we have assumed a countably infinite set of codevectors. In this section, we will show that the distribution of the quantizer Q is actually discrete and takes values from a finite set of $K(T)$ codevectors which we call "effective codevectors." Both the number and the locations of the codevectors will depend on the value of the temperature parameter T. These effective codevectors are the only parameters that an algorithmic implementation will need to store in memory.

First, notice that as $T\to\infty$, (18) yields uniform association probabilities $p(\mu_i|x)=p(\mu_j|x)\forall i,j\forall x$. As a result of (26), all codevectors are located at the same point:

$$\mu_i=\mathbb{E}\left[X\right]\quad\forall i$$

which means that there is one unique effective codevector given by $\mathbb{E}[X]$.

As T is lowered below a critical value, a bifurcation phenomenon occurs, when the number of effective codevectors increases, which is a physical analogy with chemical annealing processes. Mathematically, it occurs when the existing solution μ^* given by (26) is no longer the minimum of the free energy F^*, as the temperature T crosses a critical value. Following principles from variational calculus, we can rewrite the necessary condition for optimality (22) as

$$\frac{d}{d\epsilon}F^*(\mu+\epsilon\psi)|_{\epsilon=0}=0\tag{28}$$

with the second-order condition being

$$\frac{d^2}{d\epsilon^2}F^*(\{\mu+\epsilon\psi\})|_{\epsilon=0}\geq0\tag{29}$$

for all choices of finite perturbations $\{\psi\}$. Here, we will denote by $\{y := \mu + \epsilon\psi\}$ a perturbed codebook, where ψ are perturbation vectors applied to the codevectors μ, and $\epsilon \geq 0$ is used to scale the magnitude of the perturbation. Bifurcation occurs when equality is achieved in (29), and hence, the minimum is no longer stable.[1] These conditions are described in the following theorem.

Theorem 4: Bifurcation occurs under the following condition

$$\exists y_n \text{ s.t. } p(y_n) > 0 \text{ and } \det\left[I - T\frac{\partial^2\phi(y_n)}{\partial y_n^2}C_{x|y_n}\right] = 0 \quad (30)$$

where $C_{x|y_n} := \mathbb{E}[(x-y_n)(x-y_n)^{\mathrm{T}}|y_n]$.

Proof: From direct differentiation, the optimality condition (29) becomes

$$\sum_i p(y_i)\frac{\partial^2\phi(y_i)}{\partial y_i^2}\psi^{\mathrm{T}}\left[I - T\frac{\partial^2\phi(y_i)}{\partial y_i^2}C_{x|y_i}\right]\psi + T\int p(x)\left(\sum_i p(y_i|x)\frac{\partial^2\phi(y_i)}{\partial y_i^2}(x-y_i)^{\mathrm{T}}\psi\right)^2 dx = 0 \quad (31)$$

where

$$C_{x|y_i} := \mathbb{E}\left[(x-y_i)(x-y_i)^{\mathrm{T}}|y_i\right] = \int p(x|y_i)(x-y_i)(x-y_i)^{\mathrm{T}}dx. \quad (32)$$

The left-hand side of (31) is positive for all perturbations $\{\psi\}$ if and only if the first term is positive. To see that notice that the second term of (31) is clearly nonnegative. For the left-hand side to be nonpositive, the first term needs to be nonpositive as well, i.e., there should exist at least one codevector value, say y_n, such that $p(y_n) > 0$ and $[I - T\frac{\partial^2\phi(y_n)}{\partial y_n^2}C_{x|y_n}] \preceq 0$. In this case, there always exist a perturbation vector $\{y\}$ such that $y = 0\ \forall y \neq y_n$, and $\sum_{y=y_n}\psi = 0$, that vanishes the second term, i.e., $T\int p(x)(\sum_i p(y_i|x)\frac{\partial^2\phi(y_i)}{\partial y_i^2}(x-y_i)^{\mathrm{T}}\psi)^2 dx = 0$. In other words, we have shown that

$$\frac{d^2}{d\epsilon^2}F^*(y) > 0 \Leftrightarrow \exists y_n \text{ s.t. } p(y_n) > 0 \text{ and } \left[I - T\frac{\partial^2\phi(y_n)}{\partial y_n^2}C_{x|y_n}\right] \succ 0 \quad (33)$$

which completes the proof. ■

Notice that loss of minimality also implies that the number of effective codevectors has changed, otherwise the minimum would be stable. In addition, we have showed that bifurcation depends on the temperature coefficient T (and the choice of the Bregman divergence, through the function ϕ) and occurs when

$$\frac{1}{T} = \frac{\partial^2\phi(y_n)}{\partial y_n^2}\bar{\nu} \quad (34)$$

where $\bar{\nu}$ is the largest eigenvalue of $C_{x|y_n}$. As a result, the following corollary holds.

Corollary 4.1: The number of effective codevectors always remains bounded between two critical temperature values.

In other words, an algorithmic implementation needs only as many codevectors as the number of effective codevectors, which depends only on the changes of the temperature parameter below certain thresholds that depend on the dataset at hand and the dissimilarity measure used. As shown in Algorithm 1, we can detect the bifurcation points by introducing perturbing pairs of codevectors at each temperature level T. In this way, the codevectors μ are doubled by inserting a perturbation of each μ_i in the set of effective codevectors. The newly inserted codevectors will merge with their pair if a critical temperature has not been reached and separate otherwise. For more details about the implementation of the algorithm, the reader is referred to [12].

D. Online Learning Rule

The conditional expectation $\mathbb{E}[X|\mu]$ in (26) can be approximated by the sample mean of the data points weighted by their association probabilities $p(\mu|x)$, i.e., $\hat{\mathbb{E}}[X|\mu] = \frac{\sum xp(\mu|x)}{p(\mu)}$. This approach, however, defines an offline (batch) optimization algorithm and requires the entire dataset to be available a priori, subtly assuming that it is possible to store and also quickly access the entire dataset at each iteration. This is rarely the case in practical applications and results in computationally costly iterations that are slow to converge. We propose an ODA algorithm that dynamically updates its estimate of the effective codevectors with every observation. This results in a significant reduction in complexity that comes in two levels. The first level refers to a huge reduction in memory complexity, since we bypass the need to store the entire dataset, as well as the association probabilities $\{p(\mu_i|x)\forall x, i\}$ that map each data point in the dataset to each cluster. The second level refers to the nature of the optimization iterations. In the online approach, the optimization iterations increase in number but become much faster, and practical convergence is often reached after a smaller number of observations. To define an online training rule for the DA framework, we formulate a stochastic approximation algorithm to recursively estimate $\mathbb{E}[X|\mu]$ directly. The following theorem provides a means toward constructing a gradient-free stochastic approximation training rule for the ODA algorithm.

Theorem 5: Let S be a vector space, $\mu \in S$, and $X : \Omega \to S$ be a random variable defined in a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Let $\{x_n\}$ be a sequence of independent realizations of X, and $\{\alpha(n) > 0\}$ be a sequence of stepsizes such that $\sum_n \alpha(n) = \infty$, and $\sum_n \alpha^2(n) < \infty$. Then, the random variable $m_n = \sigma_n/\rho_n$, where (ρ_n, σ_n) are sequences defined by

$$\rho_{n+1} = \rho_n + \alpha(n)\left[p(\mu|x_n) - \rho_n\right]$$
$$\sigma_{n+1} = \sigma_n + \alpha(n)\left[x_n p(\mu|x_n) - \sigma_n\right] \quad (35)$$

converges to $\mathbb{E}[X|\mu]$ a.s., i.e., $m_n \xrightarrow{a.s.} \mathbb{E}[X|\mu]$.

Proof: We will use the facts that $p(\mu) = \mathbb{E}[p(\mu|x)]$ and $\mathbb{E}[\mathbb{1}_{[\mu]}X] = \mathbb{E}[xp(\mu|x)]$. The recursive equations (35) are

[1]For simplicity, we ignore higher order derivatives, which should be checked for mathematical completeness, but which are of minimal practical importance. The result is a necessary condition for bifurcation.

stochastic approximation algorithms of the form

$$\begin{aligned}\rho_{n+1} &= \rho_n + \alpha(n)[(p(\mu) - \rho_n) \\ &\quad + (p(\mu|x_n) - \mathbb{E}\left[p(\mu|X)\right])] \\ \sigma_{n+1} &= \sigma_n + \alpha(n)[(\mathbb{E}\left[\mathbb{1}_{[\mu]}X\right] - \sigma_n) \\ &\quad + (x_n p(\mu|x_n) - \mathbb{E}\left[x_n p(\mu|X)\right])]. \end{aligned} \tag{36}$$

It is obvious that both stochastic approximation algorithms satisfy the conditions of Theorem 1. As a result, they converge to the asymptotic solution of the differential equations

$$\begin{aligned}\dot{\rho} &= p(\mu) - \rho \\ \dot{\sigma} &= \mathbb{E}\left[\mathbb{1}_{[\mu]}X\right] - \sigma\end{aligned}$$

which can be trivially derived through standard ODE analysis to be $(p(\mu), \mathbb{E}[\mathbb{1}_{[\mu]}X])$. This follows from the fact that the only internally chain transitive invariant sets for (36) are the isolated equilibrium points $(p(\mu), \mathbb{E}[\mathbb{1}_{[\mu]}X])$. In other words, we have shown that

$$(\rho_n, \sigma_n) \xrightarrow{a.s.} \left(p(\mu), \mathbb{E}\left[\mathbb{1}_{[\mu]}X\right]\right). \tag{37}$$

The convergence of m_n follows from the fact that $\mathbb{E}[X|\mu] = {}^{\mathbb{E}[\mathbb{1}_{[\mu]}X]}/_{p(\mu)}$, and standard results on the convergence of the product of two random variables. ■

As a direct consequence of this theorem, the following corollary provides an online learning rule that solves the optimization problem of the DA algorithm.

Corollary 5.1: The online training rule

$$\begin{cases}\rho_i(n+1) &= \rho_i(n) + \alpha(n)\left[\hat{p}(\mu_i|x_n) - \rho_i(n)\right] \\ \sigma_i(n+1) &= \sigma_i(n) + \alpha(n)\left[x_n\hat{p}(\mu_i|x_n) - \sigma_i(n)\right]\end{cases} \tag{38}$$

where the quantities $\hat{p}(\mu_i|x_n)$ and $\mu_i(n)$ are recursively updated as follows:

$$\begin{aligned}\hat{p}(\mu_i|x_n) &= \frac{\rho_i(n)e^{-\frac{d(x_n,\mu_i(n))}{T}}}{\sum_i \rho_i(n)e^{-\frac{d(x_n,\mu_i(n))}{T}}} \\ \mu_i(n) &= \frac{\sigma_i(n)}{\rho_i(n)}\end{aligned} \tag{39}$$

converges a.s. to a possibly sample-path-dependent solution of the optimization (25), as $n \to \infty$.

Finally, the determinsitic annealing algorithm with the learning rule (38), (39) can be used to define a consistent (histogram) density estimator at the limit $T \to 0$. In the limit $\lambda \to 0$, and as the number of observed samples $\{x_n\}$ goes to infinity, i.e., $n \to \infty$, the learning algorithm based on (38) and (39) results in a codevector μ that constructs a consistent density estimator. This follows from the fact that as $T \to 0$, we get $p^*(\mu_i|x) \to \mathbb{1}_{[x\in S_i]}$ and $K \to \infty$, i.e., the number of effective codevectors K goes to infinity. As a result, it can be shown that $Vol(S_i) \to 0$, where $S_i = \{x \in S : i = \arg\min_j d(x, \mu_j)\}$. Then, $\hat{p}(x) = \frac{\sum_i \mathbb{1}_{[x\in S_i]}}{nVol(S_i)}$ is a consistent density estimator. The proof follows similar arguments to the stochastic vector quantization (sVQ) algorithm (see, e.g., [32]) but is omitted due to space limitations.

E. ODA for Supervised Learning

The same learning algorithm can be extended for classification as well. A multiclass classification problem involves a pair of random variables $\{X, c\} \in S \times S_c$ defined in a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, with $c \in S_c$ representing the class of X and $S \subseteq \mathbb{R}^d$. The codebook is represented by $\mu := \{\mu_i\}_{i=1}^K$, $\mu_i \in ri(S)$, and $c_\mu := \{c_{\mu_i}\}_{i=1}^K$, such that $c_{\mu_i} \in S_c$ represents the class of μ_i for all $i \in \{1, \ldots, K\}$.

We can approximate the optimal solution of a minimum classification error problem by using the distortion measure

$$d^c(x, c_x, \mu, c_\mu) = \begin{cases} d(x, \mu), & c_x = c_\mu \\ 0, & c_x \neq c_\mu. \end{cases} \tag{40}$$

It is easy to see that this particular choice for the distortion measure d^c in (40) transforms the learning rule in (38) to

$$\begin{cases}\rho_i(n+1) &= \rho_i(n) + \beta(n)\left[s_i\hat{p}(\mu_i|x_n) - \rho_i(n)\right] \\ \sigma_i(n+1) &= \sigma_i(n) + \beta(n)\left[s_i x_n\hat{p}(\mu_i|x_n) - \sigma_i(n)\right]\end{cases} \tag{41}$$

where $s_i := \mathbb{1}_{[c_{\mu_i}=c]}$. As a result, this is equivalent to estimating strongly consistent class-conditional density estimators:

$$\hat{p}(x|c=j) \to \pi_j p(x|c=j), \text{ a.s.} \tag{42}$$

where $\pi_i := \mathbb{P}[c=i]$. This results in a Bayes-optimal classification scheme. The proof is beyond the scope of this article, since we will focus our attention to RL. As a side note, a practical classification rule such as the nearest-neighbor rule:

$$\hat{c}(x) = c_{\mu_{h^*}} \tag{43}$$

where $h^* = \arg\max_{\tau=1,\ldots,K} p(\mu_\tau|x)$, $h \in \{1, \ldots, K\}$, results in an easy-to-implement classifier with tight upper bound, i.e., $J_B^* \leq \hat{J}_B^* \leq 2J_B^*$, where J_B represents the optimal Bayes error (see, e.g., [32]).

F. Algorithm

The ODA algorithm for both clustering and classification is shown in Algorithm 1 and its source code is publicly available.[2] The temperature parameter T_i is reduced using the geometric series $T_{i+1} = \gamma T_i$, for $\gamma < 1$. The temperature schedule $\{T_i\}$ affects the behavior of the algorithm by introducing the following tradeoff: small steps $T_i - T_{i-1}$ are theoretically expected to give better results, i.e., not miss any bifurcation points, but larger steps provide computational benefits.

Remark 3: Notice that the temperature schedule and its values depend on the range of the domain of the data. When the input domain is not known a priori, we can use the formulation (15) and (16), substituting T_i by $T_i := \frac{1-\lambda_i}{\lambda_i}$, $\lambda_i \in [0, 1]$.

Regarding the stochastic approximation stepsizes, simple time-based learning rates of the form $\alpha_n = {}^1/_{a+bn}$, $a, b > 0$, have experimentally shown to be sufficient for fast convergence. Convergence is checked with the condition $Td_\phi(\mu_i^n, \mu_i^{n-1}) < \epsilon_c$ for a given threshold ϵ_c. This condition becomes harder as the value of T decreases. Exploring adaptive learning rates is among the authors' future research direction. The stopping criteria

[2] https://github.com/MavridisChristos/OnlineDeterministicAnnealing

$\mathrm{SC}_{\mathrm{stop}}$ can include a maximum number of codevectors $K_{\max}$ allowed, a minimum temperature $T_{\min}$ to be reached, a minimum distortion/classification error e_{target} to be reached, a maximum number of iterations $i_{\max}$ reached, and so on.

Bifurcation, at T_i, is detected by maintaining a pair of perturbed codevectors $\{\mu_j+\delta, \mu_j-\delta\}$ for each effective codevector μ_j generated by the algorithm at T_{i-1}, i.e., for $j=1\ldots, K_{i-1}$. Using arguments from variational calculus (see Section III-C), it is easy to see that, upon convergence, the perturbed codevectors will merge if a critical temperature has not been reached, and will get separated otherwise. Therefore, the cardinality of the model is at most doubled at every temperature level. These are the effective codevectors discussed in Section III-C. For classification, a perturbed codevector for each class is generated. Merging is detected by the condition $Td_\phi(\mu_j,\mu_i)<\epsilon_n$, where ϵ_n is a design parameter that acts as a regularization term for the model that controls the number of effective codevectors. These comparisons need not be in any specific order, and the worst-case number of comparisons is $N_k=\sum_{i=1}^{K-1} i$, which scales with $O(K^2)$. An additional regularization mechanism is the detection of idle codevectors, which is checked by the condition $\rho_i(n)<\epsilon_r$, where $\rho_i(n)$ can be seen as an approximation of the probability $p(\mu_i, c_{\mu_i})$. In practice, ϵ_c, ϵ_n, and ϵ_r are assigned similar values and their impact on the performance is similar to any threshold parameter that detects convergence.

The complexity of Algorithm 1 for a fixed temperature coefficient T_i is $O(N_{c_i}(2K_i)^2\,d)$, where N_{c_i} is the number of stochastic approximation iterations needed for convergence, which corresponds to the number of data samples observed, K_i is the number of codevectors of the model at temperature T_i, and d is the dimension of the input vectors, i.e., $x\in\mathbb{R}^d$. Therefore, assuming a training dataset of N samples and a temperature schedule $\{T_1=T_{\max}, T_2,\ldots,T_{N_T}=T_{\min}\}$, the worst-case complexity of Algorithm 1 becomes

$$O(N_c(2\bar{K})^2\,d)$$

where $N_c=\max_i\{N_{c_i}\}$ is an upper bound on the number of data samples observed until convergence at each temperature level, and

$$N_T\le\bar{K}\le\min\left\{\sum_{n=0}^{N_T-1}2^n,\ \sum_{n=0}^{\log_2 K_{\max}}2^n\right\}<N_TK_{\max}$$

where the actual value of $\bar{K}$ depends on the bifurcations occurred as a result of reaching critical temperatures and the effect of the regularization mechanisms described above. Note that typically $N_c\ll N$ as a result of the stochastic approximation algorithm, and $\bar{K}\ll N_TK_{\max}$ as a result of the progressive nature of the ODA algorithm.

As a final note, because the convergence to the Bayes decision surface comes in the limit $(K,T)\to(\infty,0)$, in practice, a fine-tuning mechanism can be designed to run on top of Algorithm 1 after a predefined threshold temperature $T_{\min}$. This can be either an LVQ algorithm [33] or some other local model, i.e., we can use the partition created by Algorithm 1 to train local models in each region of the data space.

Algorithm 1: Online Deterministic Annealing (ODA).

Select a Bregman divergence d_ϕ
Set stopping criteria $\mathrm{SC}_{\mathrm{stop}}$ (e.g., $K_{\max}$, $T_{\min}$)
Set convergence parameters: γ, $\{\alpha_n\}$, ϵ_c, ϵ_n, ϵ_r, δ
Initialize: $K=1$, $\lambda=1-\epsilon$, $T=1-\lambda/\lambda$, $\{\mu^i\}$, $\{c_{\mu^i}=c,\ \forall c\in\mathcal{C}\}$, $\{p(\mu^i)=1\}$, $\{\sigma(\mu^i)=\mu^i p(\mu_i)\}$
repeat
  Perturb $\mu^i\leftarrow\{\mu^i+\delta,\mu^i-\delta\}$, $\forall i$
  Update $K\leftarrow 2K$, $p(\mu^i)$, $\sigma(\mu^i)\leftarrow\mu^i p(\mu^i)$, $\forall i$
  $n\leftarrow 0$
  repeat
    Observe data point x and class label c
    for $i=1,\ldots,K$ **do**
      Compute membership $s^i=\mathbb{1}_{[c_{\mu^i}=c]}$
      Update:

$$p(\mu^i|x)\leftarrow\frac{p(\mu^i)e^{-\frac{d_\phi(x,\mu^i)}{T}}}{\sum_j p(\mu^j)e^{-\frac{d_\phi(x,\mu^j)}{T}}}$$

$$p(\mu^i)\leftarrow p(\mu^i)+\alpha_n\left[s^i p(\mu^i|x)-p(\mu^i)\right]$$

$$\sigma(\mu^i)\leftarrow\sigma(\mu^i)+\alpha_n\left[s^i x p(\mu^i|x)-\sigma(\mu^i)\right]$$

$$\mu^i\leftarrow\frac{\sigma(\mu^i)}{p(\mu^i)}$$

      $n\leftarrow n+1$
    end for
  until Convergence: $Td_\phi(\mu^i_n,\mu^i_{n-1})<\epsilon_c$, $\forall i$
  Keep effective codevectors: discard μ^i if $Td_\phi(\mu^j,\mu^i)<\epsilon_n$, $\forall i,j,i\neq j$
  Remove idle codevectors: discard μ^i if $p(\mu^i)<\epsilon_r$, $\forall i$
  Update K, $p(\mu^i)$, $\sigma(\mu^i)$, $\forall i$
  Lower temperature: $\lambda\leftarrow\gamma\lambda$, $T\leftarrow\frac{1-\lambda}{\lambda}$
until $\mathrm{SC}_{\mathrm{stop}}$

IV. ODA FOR RL

The learning architecture of Algorithm 1 can also be integrated with RL methods, giving rise to adaptive state–action aggregation schemes. As will be shown in this section, this is a result of using stochastic approximation as a training rule, and yields an RL algorithm based on a progressively changing underlying model [34], [35].

We consider an MDP $(\mathcal{X},\mathcal{U},\mathcal{P},C)$, where $S\subseteq\mathbb{R}^{d_X}$, $S\subseteq\mathbb{R}^{d_U}$ are compact convex sets (see Section II-C). We are interested in the approximation of the quality function $Q:\mathcal{X}\times\mathcal{U}\to\mathbb{R}_+$. To this end, we use the ODA algorithm (see Algorithm 1) as an online recursive algorithm that finds an optimal representation of the data space with respect to a tradeoff between minimum average distortion and maximum entropy. We define a quantizer $Q_P(x,u)=\sum_{h=1}^K\mu_h\mathbb{1}_{[(x,u)\in P_h]}$, where $\{P_h\}_{h=1}^K$ is a partition of $\mathcal{X}\times\mathcal{U}$. The parameters $\mu_h:=(m_h,v_h)$ define a state–action aggregation scheme with K clusters (aggregate state–action pairs), each represented by $m_h\in\mathcal{X}$ and $v_h\in\mathcal{U}$, for $h=1,\ldots,K$. After convergence, if the representation is

Algorithm 2: Reinforcement Learning with ODA.

Initialize $\mu_h, Q_0(h), \forall h \in \{1, \dots, K\}$
repeat
Observe x and find

$$h = \arg\min_{\tau=1,\dots,k} d_\phi((x,u'),\mu_\tau)$$

Choose $u' = \pi_L(h|\mu)$
Observe $x' = f(x,u')$ and find

$$h' = \arg\min_{\tau=1,\dots,k} d_\phi(x,'\mu(\tau))$$

Update $Q(h)$:

$$Q_{i+1}(h) = Q_i(h) + \alpha_i[C(x,u') + \gamma \min_u Q_i(h') - Q_i(h)]$$

if $i \mod N = 0$ **then**
Update partition $\mu := \{\mu_h\}_{h=1}^K$ using Algorithm 1
end if
until Convergence
Update Policy:

$$u^*(x) = \arg\min_u \{ Q(h(x)) \}$$

meaningful, the finite set $\{\mu_h\}_{h=1}^K$, where $\mu_h \in \mathcal{X} \times \mathcal{U}$, can be used directly as a piecewise-constant approximation of the Q function. We stress that the cardinality K of the set of representatives of the space $\mathcal{X} \times \mathcal{U}$ is automatically updated by Algorithm 1 and progressively increases, as needed, with respect to the complexity–accuracy tradeoff presented above.

Remark 4: It is also possible to use $\{\mu_h\}_{h=1}^K$ as pseudoinputs for an adaptive and sparse Gaussian process regression [36], but this is beyond the scope of this article.

In essence, we are approximating the Q function with a piecewise-constant parametric model with the parameters that define the partition living in the data space and being chosen by the ODA algorithm (see Algorithm 1). However, since the system observes its states and actions online while learning its optimal policy using a temporal-difference RL algorithm, the two estimation algorithms need to run at the same time. This can become possible by observing that Algorithm 1, as well as most temporal-difference algorithms, are stochastic approximation algorithms. Therefore, we can design an RL algorithm as a two-timescale stochastic approximation algorithm with 1) a fast component that updates the values $Q := \{Q(h)\}_{h=1}^K$ with a temporal-difference learning algorithm, and 2) a slow component that updates the representation $\mu := \{\mu_h\}_{h=1}^K$ based on Algorithm 1. Such a framework can incorporate different RL algorithms, including the proposed algorithm presented in Algorithm 2. The exploration policy $\pi_L(h|\mu)$ in Algorithm 2 depends on the aggregate state h and balances the ratio between exploration and exploitation.

The convergence properties of the algorithm can be studied by directly applying the theory of the ODE method for stochastic approximation in multiple timescales in Theorem 2. For more details, see [20]. As a result, Algorithm 2 converges according to the following theorem.

Theorem 6: Algorithm 2 converges a.s. to (μ^*, Q^*), where $\mu*$ is a solution of the optimization problem (25) and Q^* minimizes the temporal-difference error:

$$J_h = \|\mathbb{E}\left[C(x,u) + \gamma \min_u Q(h')|(x,u) \in P_h\right] - Q(h)\|^2 \tag{44}$$

where $h = 1, \dots, K$, and $\{P_h\}_{h=1}^K$ is a partition of $\mathcal{X} \times \mathcal{U}$ with every P_h assumed to be visited infinitely often.

Proof: From Theorem 5, it follows that Algorithm (1) is a stochastic approximation algorithm of the form (10) that converges to a solution of (25). It is easy to see that the Q function update in Algorithm 2 is also a stochastic approximation algorithm of the form (9), for $f(Q(h)) = -\nabla_{Q(h)} J_h$. The result follows from Theorem 2. ■

We note that the condition $\beta_i/\alpha_i \to 0$ is of great importance. Intuitively, Algorithm 2 consists of two components running in different timescales. The slow component updates μ and is viewed as quasi-static when analyzing the behavior of the fast transient Q that updates the approximation of the quality function. As an example, the condition $\beta_n/\alpha_n \to 0$ is satisfied by stepsizes of the form $(\alpha_n, \beta_n) = (1/n, 1/1+n\log n)$ or $(\alpha_n, \beta_n) = (1/n^{2/3}, 1/n)$. Another way of achieving the two-timescale effect is to run the iterations for the slow component $\{\mu_n\}$ with stepsizes $\{\alpha_{n(k)}\}$, where $n(k)$ is a subsequence of n that becomes increasingly rare (i.e., $n(k+1) - n(k) \to \infty$), while keeping its values constant between these instants. In practice, it has been observed that a good policy is to run the slow component with a slower stepsize schedule β_n and update it along a subsequence keeping its values constant in between [20, Ch. 6]. This explains the parameter N in Algorithm 2 whose value should increase with time.

Remark 5: Algorithm 2 is essentially based on successive entropy-regularized RL problems. However, the entropy is defined with respect to the learning representation in the state–action space $\mathcal{X} \times \mathcal{U}$. As such, this approach is not to be directly compared to common entropy-regularized approaches as in [37], and related methods, such as PPO [38]. The deeper connection to risk-sensitive RL can be studied along the lines of [14] and [39].

V. Experimental Evaluation and Discussion

We illustrate the properties and evaluate the performance of the proposed algorithm in supervised, unsupervised, and RL problems.

A. Supervised and Unsupervised Learning

We first illustrate the properties of Algorithm 1, in a classification problem where the data samples were sampled from a mixture of 2-D Gaussian distributions. In Figs. 1 and 2, the temperature level (the values of T shown are the normalized values $\lambda = 1/T+1$), the average distortion of the model, the number of codevectors (neurons) used, the number of observations (data samples) used for convergence, as well as the overall time are shown. Since the objective is to give a geometric illustration of how the algorithm works in the 2-D plane, the Euclidean distance

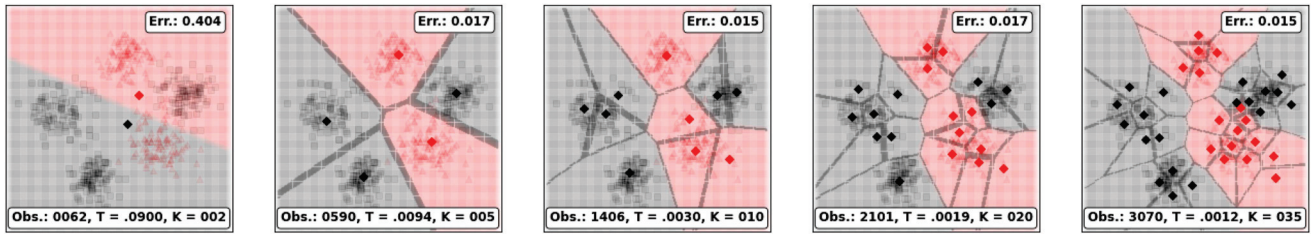

Fig. 1. Evolution of the ODA algorithm in 2-D binary classification based on class-conditional density estimation. Temperature T decreases from left to right.

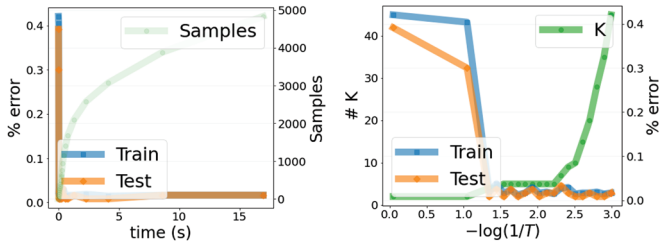

Fig. 2. Performance curves for the problem of Fig. 1.

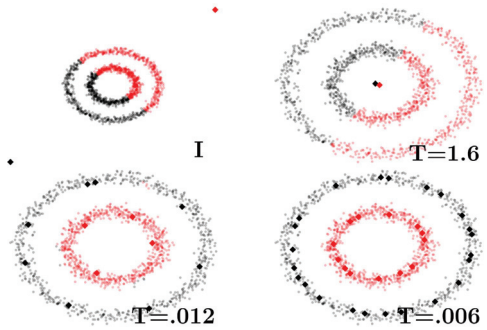

Fig. 3. Illustration of the evolution of Algorithm 1 for binary classification in 2-D based on class-conditional density estimation. Showcasing robustness to bad initial conditions, which indicates poor local minima prevention.

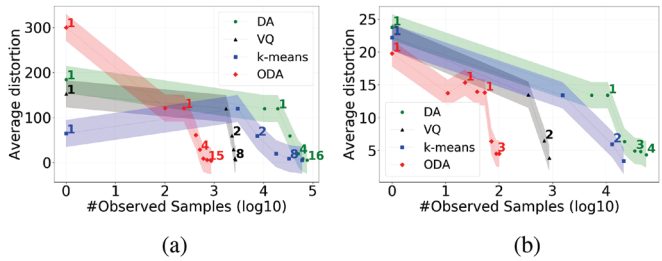

Fig. 4. Algorithm comparison for unsupervised learning. (a) Gaussians. (b) PIMA.

is used as the proximity measure. Notice that the classification accuracy for $K = 5$ is 98% and it gets to 100% only when we reach $K = 144$. This showcases the performance–complexity tradeoff that Algorithm 1 allows the user to control in an online fashion. Since classification and clustering are handled in a similar way by Algorithm 1, these examples properly illustrate the behavior of the proposed methodology for clustering as well.

In Fig. 3, the progression of the learning representation is depicted for a binary classification problem with underlying class distributions shaped as concentric circles. The algorithm starts at a high temperature with a single codevector for each class. Here, the codevectors are poorly initialized outside the support of the data, which are not assumed known a priori (e.g., online observations of unknown domain). In this example, the LVQ algorithm has been shown to fail [40]. This showcases the robustness of the proposed algorithm with respect to the initial configuration. This is an example of poor local minima prevention, which, although not theoretically guaranteed, is a known property of annealing optimization methods. For clustering, we consider the dataset of Fig. 1 (Gaussians), and the PIMA dataset [41]. In Fig. 4, we compare Algorithm 1 with the stochastic (online) vector quantization (sVQ) algorithm [40], and two offline (batch) algorithms, namely k-means [42] and the original DA algorithm [17]. The algorithms are compared in terms of the minimum average distortion achieved, as a function of the number of samples they observed and the number of clusters they used. The Euclidean distance is used for fair comparison. Since there is no criterion to decide the number of clusters K for k-means and sVQ, we run them sequentially for the K values estimated by DA, and add up the computational time. All algorithms are able to achieve comparable average distortion values, given good initial conditions and appropriate size K. Therefore, the progressive estimation of K and the robustness with respect to the initial conditions are key features of both annealing algorithms. Compared to the offline algorithms, i.e., k-means and DA, ODA, and sVQ achieve practical convergence with a significantly lower number of observations, which corresponds to reduced computational time, as argued above. Compared to the online sVQ (and LVQ), the probabilistic approach of ODA introduces additional computational cost: all neurons are now updated in every iteration, instead of only the winner neuron. However, the updates can still be computed relatively fast when using Bregman divergences (see Theorem 3). For more experimental results regarding clustering and classification, the readers are referred to [12], [36], and [43].

B. RL

Finally, we validate the proposed methodology on the inverted pendulum (cart–pole) optimal control problem. The state

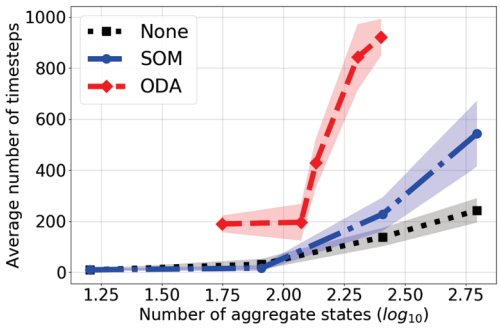

Fig. 5. Average number of timesteps ($N_t = 1000$) over number of aggregate states used. Red represents the proposed algorithm. Black represents Q-learning without state aggregation. Blue represents the SOM-based algorithm of [34].

variable of the cart–pole system has four components $(x, \theta, \dot{x}, \dot{\theta})$, where x and $\dot{x}$ are the position and velocity of the cart on the track, and θ and $\dot{\theta}$ are the angle and angular velocity of the pole with the vertical. The cart is free to move within the bounds of a 1-D track. The pole is free to move only in the vertical plane of the cart and track.

The action space consists of an impulsive "left" or "right" force $F \in \{-10, +10\}$ N of fixed magnitude to the cart at discrete time intervals. The cart–pole system is modeled by the following nonlinear system of differential equations [44]:

$$\ddot{x} = \frac{F + ml\left(\dot{\theta}^2 \sin\theta - \ddot{\theta}\cos\theta\right)\mu_c sgn(\dot{x})}{m_c + m}$$

$$\ddot{\theta} = \frac{g\sin\theta + \cos\theta\left(\frac{-F - ml\dot{\theta}^2\sin\theta + \mu_c sgn(\dot{x})}{m_c+m}\right) - \frac{\mu_p}{ml}\mu_c sgn(\dot{x})}{l\left(\frac{4}{3} - \frac{m\cos^2\theta}{m_c+m}\right)}$$

where the parameter values for g, m_c, m, l, μ_c, and μ_p can be found in [44]. The transition function for the state x is $x_{n+1} = x_n + \tau\dot{x}$, where $\tau = 0.02$ s. The initial state is set to $X_0 = (u_x, u_\theta, u_{\dot{x}}, u_{\dot{\theta}})$, where u_x, u_θ, $u_{\dot{x}}$, and $u_{\dot{\theta}}$ follow a uniform distribution $U(-0.05, 0.05)$. Failure occurs when $|\theta| > 12^\circ$ or when $|x| > 2.4$ m. An episode terminates successfully after N_t timesteps, and the average number of timesteps $\hat{N}_t \leq N_t$ across different attempts is used to quantify the performance of the learning algorithm. We use the Euclidean distance as the Bregman divergence d_ϕ.

In Fig. 5, we compare the average number of timesteps (here, $N_t = 1000$) with respect to the number of aggregate states used, for three different state aggregation algorithms. The first one is naive discretization without state aggregation, the second is the SOM-based algorithm proposed in [34], and the last is the proposed algorithm (see Algorithm 2). We initialize the codevectors μ by uniformly discretizing over $\hat{S} \times \{-10, 10\}$, for $\hat{S} = [-1, 1] \times [-4, 4] \times [-1, 1] \times [-4, 4]$. We use $K \in \{16, 81, 256, 625\}$ clusters, corresponding to a standard discretization scheme with only $n \in \{2, 3, 4, 5\}$ bins for each dimension. As expected, state aggregation outperforms standard discretization of the state–action space. The ability to progressively adapt the number and placement of the centroids of the aggregate states is an important property of the proposed algorithm. Fig. 5 shows five instances of Algorithm 2 for five different stopping criteria according to a predefined minimum temperature $T_{\min}$. This results in different representations of the state space with $K \in \{56, 118, 136, 202, 252\}$ aggregate states, respectively.

VI. Conclusion

We investigate the properties of learning with progressively growing models, and propose an online annealing optimization approach as a learning algorithm that progressively adjusts its complexity with respect to new observations, offering online control over the performance–complexity tradeoff. We show that the proposed algorithm constitutes a progressively growing competitive-learning neural network with inherent regularization mechanisms, the learning rule of which is formulated as an online gradient-free stochastic approximation algorithm. The use of stochastic approximation enables the study of the convergence of the learning algorithm through mathematical tools from dynamical systems and control, and allows for its use in supervised, unsupervised, and RL settings. In addition, the annealing nature of the algorithm, contributes to poor local minima prevention and offers robustness with respect to the initial conditions. To the best of our knowledge, this is the first time such a progressive approach has been proposed for machine learning and RL applications. These results can lead to new developments in the development of progressively growing machine learning models targeted toward applications in which computational resources are limited and robustness and interpretability are prioritized.

References

[1] K. P. Bennett and E. Parrado-Hernández, "The interplay of optimization and machine learning research," *J. Mach. Learn. Res.*, vol. 7, pp. 1265–1281, 2006.
[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[3] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The computational limits of deep learning," 2020, *arXiv:2007.05558*.
[4] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," *arXiv:1906.02243*.
[5] C. Szegedy et al., "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.
[6] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 39–57.
[7] V. Sehwag et al., "Analyzing the robustness of open-world machine learning," in *Proc. 12th ACM Workshop Artif. Intell. Secur.*, 2019, pp. 105–116.
[8] H. Xu and S. Mannor, "Robustness and generalization," *Mach. Learn.*, vol. 86, no. 3, pp. 391–423, 2012.
[9] C. G. Northcutt, A. Athalye, and J. Mueller, "Pervasive label errors in test sets destabilize machine learning benchmarks," *35th Conf. Neural Inf. Process. Syst. Datasets Benchmarks Track*, 2021.
[10] A. Raghunathan, S. M. Xie, F. Yang, J. C. Duchi, and P. Liang, "Adversarial training can hurt generalization," *ICML Workshop Identifying Understanding Deep Learn. Phenomena*, 2019.
[11] C. N. Mavridis and J. S. Baras, "Convergence of stochastic vector quantization and learning vector quantization with Bregman divergences," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 2214–2219, 2020.
[12] C. N. Mavridis and J. S. Baras, "Online deterministic annealing for classification and clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2021.3138676.

[13] M. Biehl, B. Hammer, and T. Villmann, "Prototype-based models in machine learning," *Wiley Interdiscipl. Rev., Cogn. Sci.*, vol. 7, no. 2, pp. 92–111, 2016.
[14] C. Mavridis, E. Noorani, and J. S. Baras, "Risk sensitivity and entropy regularization in prototype-based learning," in *Proc. IEEE 30th Mediterranean Conf. Control Automat.*, 2022, pp. 194–199.
[15] E. A. Uriarte and F. D. Martín, "Topology preservation in SOM," *Int. J. Appl. Math. Comput. Sci.*, vol. 1, no. 1, pp. 19–22, 2005.
[16] S. Saralajew, L. Holdijk, M. Rees, and T. Villmann, "Robustness of generalized learning vector quantization models against adversarial attacks," in *Proc. Int. Workshop Self-Organizing Maps*, 2019, pp. 189–199.
[17] K. Rose, "Deterministic annealing for clustering, compression, classification, regression, and related optimization problems," *Proc. IEEE*, vol. 86, no. 11, pp. 2210–2239, Nov. 1998.
[18] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.
[19] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statis.*, pp. 400–407, 1951.
[20] V. S. Borkar, *Stochastic Approximation: A Dynamical Systems Viewpoint*, vol. 48. Berlin, Germany: Springer, 2009.
[21] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh, "Clustering with Bregman divergences," *J. Mach. Learn. Res.*, vol. 6, no. Oct, pp. 1705–1749, 2005.
[22] T. Villmann, S. Haase, F.-M. Schleif, B. Hammer, and M. Biehl, "The mathematics of divergence based online learning in vector quantization," in *Proc. IAPR Workshop Artif. Neural Netw. Pattern Recognit.*, 2010, pp. 108–119.
[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.
[24] L. Bottou, "Online learning and stochastic approximations," *On-line Learn. Neural Netw.*, vol. 17, no. 9, pp. 9–42, 1998.
[25] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 279–292, 1992.
[26] J. N. Tsitsiklis, "Asynchronous stochastic approximation and q-learning," *Mach. Learn.*, vol. 16, no. 3, pp. 185–202, 1994.
[27] V. S. Borkar and S. P. Meyn, "The ODE method for convergence of stochastic approximation and reinforcement learning," *SIAM J. Control Optim.*, vol. 38, no. 2, pp. 447–469, 2000.
[28] V. S. Borkar, "Stochastic approximation with two time scales," *Syst. Control Lett.*, vol. 29, no. 5, pp. 291–294, 1997.
[29] T. Kohonen, *Learning Vector Quantization*. Berlin, Germany: Springer, 1995, pp. 175–189.
[30] E. T. Jaynes, "Information theory and statistical mechanics," *Phys. Rev.*, vol. 106, no. 4, pp. 620–630, 1957.
[31] T. D. Hocking, A. Joulin, F. Bach, and J.-P. Vert, "Clusterpath an algorithm for clustering using convex fusion penalties," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, p. 1.
[32] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, vol. 31. Berlin, Germany: Springer Sci. Bus. Media, 2013.
[33] A. Sato and K. Yamada, "Generalized learning vector quantization," in *Proc. Adv. neural Inf. Process. Syst.*, 1996, pp. 423–429.
[34] C. N. Mavridis and J. S. Baras, "Vector quantization for adaptive state aggregation in reinforcement learning," in *Proc. IEEE Amer. Control Conf.*, 2021, pp. 2187–2192.
[35] C. N. Mavridis, N. Suriyarachchi, and J. S. Baras, "Maximum-entropy progressive state aggregation for reinforcement learning," in *Proc. IEEE 60th Conf. Decis. Control*, 2021, pp. 5144–5149.
[36] C. N. Mavridis, G. Kontoudis, and J. S. Baras, "Sparse Gaussian process regression using progressively growing learning representations," in *Proc. IEEE 61st Conf. Decis. Control*, 2022, pp. 5144–5149.
[37] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1352–1361.
[38] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[39] E. Noorani, C. Mavridis, and J. Baras, "Risk-sensitive reinforcement learning with exponential criteria," 2022, *arXiv:2212.09010*.
[40] J. S. Baras and A. LaVigna, "Convergence of a neural network classifier," in *Proc. Adv. Neural Inf. Process. Syst.*, 1991, pp. 839–845.
[41] J. W. Smith, J. Everhart, W. Dickson, W. Knowler, and R. Johannes, "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus," in *Proc. Annu. Symp. Comput. Appl. Med. Care*. 1988, p. 261.
[42] L. Bottou and Y. Bengio, "Convergence properties of the k-means algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 1995, pp. 585–592.
[43] C. N. Mavridis and J. S. Baras, "Progressive graph partitioning based on information diffusion," in *Proc. IEEE 60th Conf. Decis. Control*, 2021, pp. 37–42.
[44] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, no. 5, pp. 834–846, Sep./Oct. 1983.

Christos N. Mavridis (Member, IEEE) received the diploma degree from the National Technical University of Athens, Athens, Greece, in 2017, and the M.S. and Ph.D. degrees from the University of Maryland, College Park, MD, USA, in 2021, all in electrical and computer engineering.

He was a Postdoctoral Associate with the University of Maryland, and a Visiting Postdoctoral Fellow with the KTH Royal Institute of Technology, Stockholm. He was a Research Intern for the Math and Algorithms Research Group, Nokia Bell Labs, NJ, USA, and the System Sciences Lab, Xerox Palo Alto Research Center (PARC), CA, USA. His research interests include systems and control theory, stochastic optimization, learning theory, multiagent systems, and robotics.

Dr. Mavridis is a Member of the Institute for Systems Research (ISR) and the Autonomy, Robotics and Cognition (ARC) Lab. He was the recipient of the Ann G. Wylie Dissertation Fellowship in 2021, and the A. James Clark School of Engineering Distinguished Graduate Fellowship, the Outstanding Graduate Research Assistant Award, the Future Faculty Fellowship, in 2017, 2020, and 2021, respectively, and he has received the Best Student Paper Award (1st Place) at the IEEE International Conference on Intelligent Transportation Systems (ITSC), 2021. He has been a finalist in the Qualcomm Innovation Fellowship US, San Diego, CA, 2018.

John S. Baras (Life Fellow, IEEE) received the diploma degree in electrical and mechanical engineering from the National Technical University of Athens, Athens, Greece, in 1970, and the M.S. and Ph.D. degrees in applied mathematics from Harvard University, Cambridge, MA, USA, in 1971 and 1973, respectively.

From 1985 to 1991, he was the Founding Director of the ISR. Since 1992, he has been the Director of the Maryland Center for Hybrid Networks (HYNET), which he co-founded. He is currently a Distinguished University Professor and holds the Lockheed Martin Chair in Systems Engineering with the Department of Electrical and Computer Engineering and the Institute for Systems Research (ISR), University of Maryland College Park, MD, USA. His research interests include systems and control, optimization, communication networks, applied mathematics, machine learning, artificial intelligence, signal processing, robotics, computing systems, security, trust, systems biology, healthcare systems, and model-based systems engineering.

Dr. Baras is a Fellow of SIAM, AAAS, NAI, IFAC, AMS, and AIAA, a Member of the National Academy of Inventors, and a Foreign Member of the Royal Swedish Academy of Engineering Sciences. He was the recipient of major honors, including the 1980 George Axelby Award from the IEEE Control Systems Society, the 2006 Leonard Abraham Prize from the IEEE Communications Society, the 2017 IEEE Simon Ramo Medal, the 2017 AACC Richard E. Bellman Control Heritage Award, the 2018 AIAA Aerospace Communications Award. In 2016, he was inducted in the A. J. Clark School of Engineering Innovation Hall of Fame. In 2018, he was awarded a Doctorate Honoris Causa by his alma mater the National Technical University of Athens, Greece.