Explainable Learning with Hierarchical Online Deterministic Annealing*

Christos N. Mavridis[0000−0001−9612−8903] and John S. Baras[0000−0002−4955−8561]

University of Maryland, MD 20742, USA
{mavridis,baras}@umd.edu

Abstract. We introduce a general-purpose hierarchical approximation algorithm based on the principles of online deterministic annealing, showcasing its properties in the context of explainable machine learning. The main idea is to progressively construct a partition of the data space using gradient-free stochastic approximation updates and use local learning models that can be trained online using two-timescale stochastic approximation. As the partition adapts to the data space at hand, a progressively more detailed representation of the the data space is constructed in the form of a hierarchically structured set of regions. Mathematically, this is a tree-structured partition in a multi-resolution representation of the data space. As a result, the complexity of the local models is greatly reduced and common problems such as over-fitting and poor local minima can be mitigated. In addition, this process introduces hierarchical variable-rate feature extraction properties similar to certain classes of deep learning architectures. Experimental results for supervised learning problems illustrate the properties of the proposed method as an explainable machine learning algorithm.

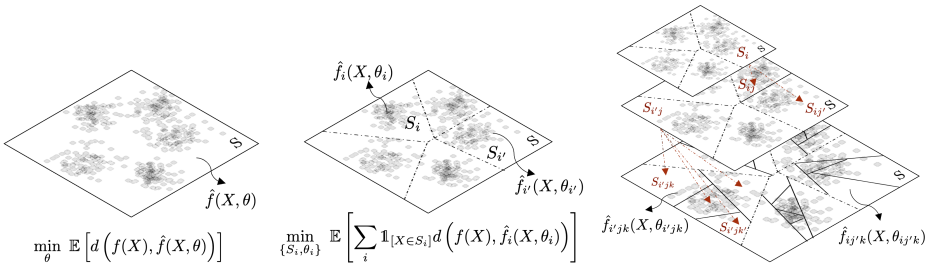
Keywords: Explainable AI · Online Deterministic Annealing · Hierarchical Learning · Hierarchical Clustering · Stochastic Approximation.

1 Introduction

While recent deep learning methods have shown experimental success as complex general function approximation models over the entire data space [12,15,17,18], there are still open problems regarding the time, energy, data, memory, and computational cost of the estimation of θ [33,34], the phenomena of robustness, over-fitting, and poor local minima [4], and the explainability properties of these models [13,31]. In particular, the use of complex black-box models inherently hinders the ability to understand the properties of the data space, information that can be used to enhance the performance of the optimization algorithms, reduce their complexity, improve their robustness with respect to noise and adversarial attacks, and support lifelong learning [31].

* Research partially supported by ONR grant N00014-17-1-2622, and by a grant from Northrop Grumman Corporation.

In this work, we focus on a framework for hierarchical progressive learning and data representation, where a gradually growing and hierarchically structured set of learning models is used for function approximation. We consider a prototype-based learning framework where a set of prototypes (also called codevectors or neurons) are scattered in the data space S to encode subsets/regions $\{S_i\}$ that form a partition of S [5]. This adheres to the principles of vector quantization for signal compression [14]. In this regard, a knowledge representation can be defined as a set of codevectors $\{\mu_i \in S\}$ that induce a structured partition $\{S_i\}$ of the data space, along with a set of local learning models $\hat{f}(x, \theta_i)$ associated with each region S_i, parameterized by their own set of parameters θ_i. A structured representation like this allows, among other things, to locate regions of the space that the algorithm needs to approximate in greater detail, according to the problem at hand and the designer's requirements. This results in adaptively allocating more resources in the subsets of the data space that are needed, and provides benefits in terms of time, memory, and model complexity [22]. Moreover, learning with local models that take advantage of the differences in the underlying distribution of the data space provides a means to understand certain properties of the data space itself, i.e., this is an interpretable learning approach [32]. An illustration of this framework is given in Fig. 1.



(a) Classical regression problem.

(b) Combined problem of partitioning and function approximation.

(c) Hierarchical partitioning and function approximation.

Fig. 1: Comparison of the classical regression problem over the entire data space S with the problem of combined partitioning and regression.

Regarding the learning process, we are interested in algorithms that are able to simultaneously solve both the problems of partitioning and function approximation, given online (e.g., real-time) observations. This is of great importance in many applications, and especially in the scope of learning algorithms for inference and control in general cyber-physical systems [23, 26, 29]. To construct a sequence of partitions with increasing number of subsets we build upon the notion of Online Deterministic Annealing [27]. One important property of this approach is that, by successively solving a sequence of optimization problems, a series of bifurcation phenomena arises when the cardinality of the set of code-

vectors $\{\mu_i\}$ increases as needed.A second important property of this approach, initially shown in [27], is that the optimization problems can be solved online using gradient-free stochastic approximation updates [7]. We exploit the fact that a stochastic approximation algorithm can be used as a training rule for constructing the partition $\{S_i\}$, to build a framework that simultaneously trains the learning models $\left\{\hat{f}(x,\theta_i)\right\}$ defined in each region S_i. In particular, according to the theory of two-timescale stochastic approximation [7], we define two stochastic approximation algorithms that run at the same time and with the same observations but with different stepsize schedules that define a fast and a slow learing process. In our case the slow process approximates the parameters $\{\mu_i\}$ and as a result the partition $\{S_i\}$, and the fast process executes a function approximation algorithm within each S_i to find the optimal parameters θ_i for the learning model $\hat{f}(x,\theta_i)$.

Finally, by imposing a non-binary tree structure in the growing set of the parameters $\{\mu_i\}$, we show that we can both (a) greatly reduce the quadratic (in the number of parameters μ_i) complexity of the approach, and (b) construct a hierarchical and progressively growing tree-structured partition where each layer of the tree is trained using different resolution representation of the data space, according to an independent multi-resolution analysis. While this is a general framework for multi-resolution learning, we show that, in the case when convolution-based multi-resolution features are used, the proposed architecture shares similarities with deep learning approaches such as Deep Convolutional Networks [17] and Scattering Convolutional Networks [9]. Experimental results illustrate the properties of the proposed approach in clustering, classification, and regression applications.

2 Online Deterministic Annealing

We start our analysis with the case of unsupervised learning, where partitioning a space S is equivalent to the problem of clustering and density estimation. In this context, the observations (data) are independent realization of 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). In the Online Deterministic Annealing approach [23,27], one defines a similarity measure $d : S \to ri(S)$[1], and a discrete random variable $Q : S \to ri(S)$ with domain $\mu := \{\mu_i\}_{i=1}^K$, $\mu_i \in ri(S)$ described by the association probabilities $\{p(\mu_i|x) := \mathbb{P}[Q = \mu_i|X = x]\}$, $\forall i$, such that

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

This is a multi-objective optimization problem formulated as the minimization of a Lagrangian function, where $\lambda \in [0,1)$ acts as a Lagrange multiplier controlling the trade-off between the average distortion:

$$\min_{\mu} \; D(\mu) := \mathbb{E}\left[d\left(X,Q\right)\right] = \mathbb{E}\left[\mathbb{E}\left[d(X,Q)|X\right]\right] = \int p(x) \sum_i p(\mu_i|x) d(x,\mu_i) \; dx$$

[1] $ri(S)$ represents the relative interior of S.

and the entropy term:

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

The entropy H, acts as a regularization term, and is given progressively less weight as λ decreases. The term $T := \frac{\lambda}{1-\lambda},\ \lambda \in [0,1)$ can be seen as a temperature coefficient in a deterministic annealing process [27].

Following the Online Deterministic Annealing (ODA) approach [23, 27], we minimize F_λ in (1) by successively minimizing it first respect to the association probabilities $\{p(\mu_i|x)\}$, and then with respect to the codevector locations μ. The solution of the optimization problem

$$F_\lambda^*(\mu) := \min_{\{p(\mu_i|x)\}} F_\lambda(\mu) \quad \text{s.t.} \quad \sum_i p(\mu_i|x) = 1, \tag{2}$$

is given by the Gibbs distributions $p^*(\mu_i|x) = \frac{e^{-\frac{1-\lambda}{\lambda} d(x,\mu_i)}}{\sum_j e^{-\frac{1-\lambda}{\lambda} d(x,\mu_j)}},\ \forall x \in S.$

Furthermore, it has been shown in [27] that if $d := d_\phi$ is a Bregman divergence[2], then the conditional expectation:

$$\mu_i^* = \mathbb{E}\left[X|\mu_i\right] = \frac{\int x p(x) p^*(\mu_i|x)\ dx}{p^*(\mu_i)} \tag{3}$$

is a solution to the optimization problem

$$\min_\mu F_\lambda^*(\mu). \tag{4}$$

Moreover, a stochastic approximation algorithm can be formulated [27] to recursively estimate $\mathbb{E}\left[X|\mu_i\right]$ directly, according to Theorem 1.

Theorem 1 ([27]). *Let $\{x_n\}$ be a sequence of independent realizations of X. Then $\mu_i(n)$, defined by the online training rule*

$$\begin{cases} \rho_i(n+1) & = \rho_i(n) + \alpha(n)\left[\hat{p}(\mu_i|x_n) - \rho_i(n)\right] \\ \sigma_i(n+1) & = \sigma_i(n) + \alpha(n)\left[x_n \hat{p}(\mu_i|x_n) - \sigma_i(n)\right] \end{cases} \tag{5}$$

where $\sum_n \alpha(n) = \infty$, $\sum_n \alpha^2(n) < \infty$, and the quantities $\hat{p}(\mu_i|x_n)$ and $\mu_i(n)$ are recursively updated as follows:

$$\mu_i(n) = \frac{\sigma_i(n)}{\rho_i(n)}, \quad \hat{p}(\mu_i|x_n) = \frac{\rho_i(n) e^{-\frac{1-\lambda}{\lambda} d(x_n, \mu_i(n))}}{\sum_i \rho_i(n) e^{-\frac{1-\lambda}{\lambda} d(x_n, \mu_i(n))}} \tag{6}$$

converges almost surely to a locally asymptotically stable solution of the optimization (4), as $n \to \infty$.

[2] The function d_ϕ is a Bregman divergence if there exists a strictly convex function ϕ such that $d_\phi(x,\mu) = \phi(x) - \phi(\mu) - \frac{\partial \phi}{\partial \mu}(\mu)(x-\mu)$. Two notable examples are the squared Euclidean distance $d_\phi(x,\mu) = \|x-\mu\|^2$ $(\phi(x) = \langle x, x\rangle,\ x \in \mathbb{R}^d)$, and the the generalized Kullback-Leibler divergence $d_\phi(x,\mu) = \langle x, \log x - \log \mu\rangle - \langle \mathbb{1}, x-\mu\rangle$ $(\phi(x) = \langle x, \log x\rangle,\ x \in \mathbb{R}^d_{++})$. For more details see, e.g., [3, 23, 25].

The learning rule (5), (6) is a stochastic approximation algorithm [7]. In addition, it is a discrete-time dynamical system that presents bifurcation phenomena with respect to the parameter λ, i.e., the number of equilibria of this system changes with respect to the value λ. Finally, in the limit $\lambda \to 0$, it results in a consistent density estimator, i.e., the representation of the random variable $X \in S$ by the codevectors μ becomes all the more accurate in S, according to the underlying probability density $p(x)$ [23].

2.1 Bifurcation, Algorithmic Implementation, and Complexity

So far, we have assumed a countably infinite set of codevectors. In this section we will show that the unique values of the set $\{\mu_i\}$ that solves (1), form a finite set $K(\lambda)$ of values that we will refer to as "effective codevectors". These effective codevectors are the only values that an algorithmic implementation will need to store in memory and update. First, notice that when $\lambda \to 1$ (resp. $T \to \infty$) equation (2) yields uniform association probabilities $p(\mu_i|x) = p(\mu_j|x),\ \forall i, j, \forall x$, such that all codevectors are located at the same point $\mu_i = \mathbb{E}[X],\ \forall i$, meaning that there is one unique effective codevector given by $\mathbb{E}[X]$. As λ is lowered below a critical value, a bifurcation phenomenon occurs, when the number of effective codevectors, i.e., the number of equillibria of (5), (6) increases. Following principles from variational calculus, we can detect the critical values of λ when bifurcation occurs [23]. However, as shown in Alg. 1, in practice we detect the bifurcation points by introducing perturbing pairs of codevectors at each temperature level λ. In this way, the codevectors μ are doubled and the newly inserted codevectors will merge with their pair if a critical temperature has not been reached and separate otherwise [27].

The complexity of Alg. 1 for fixed coefficient λ_t is $O(N_{c_t}(2K_t)^2 d)$, where N_{c_t} is the number of stochastic approximation iterations needed for convergence which corresponds to the number of data samples observed, K_t is the number of codevectors of the model at temperature λ_t, and d is the dimension of the input vectors, i.e., $x \in \mathbb{R}^d$. Therefore, assuming a coefficient schedule $\{\lambda_1 = \lambda_{max}, \lambda_2, \ldots, \lambda_{N_\lambda} = \lambda_{min}\}$, the time complexity for the training of Algorithm 1 becomes: $O(N_c(2\bar{K})^2 d)$, where $N_c = \max_i \{N_{c_t}\}$ is an upper bound on the number of data samples observed until convergence at each temperature level, and $\bar{K} = \sum_{i=1}^{N_\lambda} K_t$, with $N_\lambda \leq \bar{K} \leq \min\left\{\sum_{n=0}^{N_\lambda - 1} 2^n, \sum_{n=0}^{\log_2 K_{max}} 2^n\right\} < N_\lambda K_{max}$. 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 $\bar{K} \ll N_\lambda K_{max}$ as a result of the progressive nature of the algorithm. Prediction scales linearly with $O(K_{N_\lambda} d)$, with $K_{N_\lambda} \leq K_{max}$.

3 Learning with Local Models

In this section, we investigate the problem of combined partitioning and function approximation, where multiple local models are trained, taking advantage of the

Algorithm 1 Progressive Partitioning.

Select a Bregman divergence d_ϕ
Set stopping criteria T_{stop} (e.g., K_{max}, λ_{min})
Set convergence parameters: γ, ϵ_c, ϵ_n, ϵ_r, δ
Set stepsizes: $\{\alpha_n\}$
Initialize: $K = 1$, $\lambda = 1$,
$\{\mu_0\}$, $p(\mu_0) = 1$, $\sigma(\mu_0) = \mu_0 p(\mu_0)$
repeat
  Perturb codebook: $\{\mu_i\} \leftarrow \{\mu_i + \delta\} \bigcup \{\mu_i - \delta\}$
  Update $K \leftarrow 2K$, $\{p(\mu_i)\}$, $\{\sigma(\mu_i) \leftarrow \mu_i p(\mu_i)\}$
  $n \leftarrow 0$
  repeat
    Observe data point x
    for $i = 1, \ldots, K$ **do**
      Update:

$$p(\mu_i|x) \leftarrow \frac{p(\mu_i) e^{-\frac{1-\lambda}{\lambda} d_\phi(x,\mu_i)}}{\sum_i p(\mu_i) e^{-\frac{1-\lambda}{\lambda} d_\phi(x,\mu_i)}}$$

$$p(\mu_i) \leftarrow p(\mu_i) + \alpha_n \left[p(\mu_i|x) - p(\mu_i)\right]$$

$$\sigma(\mu_i) \leftarrow \sigma(\mu_i) + \alpha_n \left[x p(\mu_i|x) - \sigma(\mu_i)\right]$$

$$\mu_i \leftarrow \frac{\sigma(\mu_i)}{p(\mu_i)}$$

    $n \leftarrow n + 1$
    end for
  until Convergence: $\frac{1-\lambda}{\lambda} d_\phi(\mu_i^n, \mu_i^{n-1}) < \epsilon_c$, $\forall i$
  Keep effective codevectors:
    discard μ_i if $\frac{1-\lambda}{\lambda} d_\phi(\mu_j, \mu_i) < \epsilon_n$, $\forall i, j, i \neq j$
  Remove idle codevectors:
    discard μ_i if $p(\mu_i) < \epsilon_r$, $\forall i$
  Update K, $\{p(\mu_i)\}$, $\{\sigma(\mu_i)\}$
  Lower temperature: $\lambda \leftarrow \gamma\lambda$
until T_{stop}

differences in the underlying probability distribution of the data space. As a consequence, this approach can circumvent the use of overly complex learning models, reduce time, memory, and computational complexity, and give insights to certain properties of the data space [32].

Assuming models $\hat{f}_i(x, \theta_i) \in \mathcal{F}$ that are differentiable with respect to a parameter vector $\theta_i \in \Theta$, where Θ is a finite-dimensional vector space, and given a finite partition set of parameters $\{S_i\}_{i=1}^{K(\lambda)}$, for $K(\lambda) < \infty$, the problem is formulated as:

$$\min_{\theta_i} \; \mathbb{E}\left[\mathbb{1}_{[X \in S_i]} d\left(f(X), \hat{f}_i(X, \theta_i)\right)\right], \; i = 1, \ldots, K(\lambda). \tag{7}$$

where $d : \mathcal{F} \times \mathcal{F} \to [0, \infty)$ is assumed a metric that is differentiable and convex with respect to the second argument. This is a stochastic optimization problem that can be solved using stochastic approximation updates. In particular, one

can use stochastic gradient descent:

$$\theta_i(n+1) = \theta_i(n) - \beta(n)\nabla_\theta d(f(x_n), \hat{f}_i(x_n, \theta_i(n))) \tag{8}$$

which is a special case of a stochastic approximation algorithm that, under mild assumptions, converges almost surely to an asymptotically stable local minimum of the objective function $\mathbb{E}\left[\mathbb{1}_{[X\in S_i]}\hat{d}(f(x_n), f_i(x, \theta_i))\right]$.

However, we are interested in a learning approach that approximates $\{S_i\}$ and $\left\{\hat{f}_i(x, \theta_i)\right\}$ at the same time, and given the same observations $\{(x_n, f(x_n))\}$ which may be available one at a time (i.e, no dataset is stored in memory a priori). This is possible because both learning algorithms for $\{S_i\}$ and $\left\{\hat{f}_i(x, \theta_i)\right\}$ independently are stochastic approximation algorithms. According to the theory of two-timescale stochastic approximation, we can run both learning algorithms at the same time, but using different stepsize profiles $\{\alpha(n)\}$ and $\{\beta(n)\}$, such that $\alpha(n)/\beta(n) \to 0$. Intuitively, we create a system of two dynamical system running in different "speed", meaning that second system, the one with stepsizes $\{\beta(n)\}$, is updated fast enough that the first system, the one with stepsizes $\{\alpha(n)\}$, can be seen as quasi-static with respect to the second. The following theorem follows directly from the results of Ch. 6 in [6].

Theorem 2. *Let $\{x_n\}$ be a sequence of independent realizations of X, and assume that $\mu_i(n)$ is a sequence updated using the stochastic approximation algorithm in (5) with stepsizes $\{\alpha(n)\}$ satisfying $\sum_n \alpha(n) = \infty$, and $\sum_n \alpha^2(n) < \infty$. Then, as long as $\{\beta(n)\}$ are designed such that $\sum_n \beta(n) = \infty$, $\sum_n \beta^2(n) < \infty$, and $\alpha(n)/\beta(n) \to 0$, the asynchronous updates*

$$\theta_i(n+1) = \theta_i(n) - \beta(n)\nabla_\theta d(f(x_n), \hat{f}_i(x_n, \theta_i(n))), \tag{9}$$

for $i = \arg\min_j \; d_\phi(x_n, \mu_j(n))$ converges almost surely to a locally asymptotically stable solution $\{\theta_i\}$ of (7), as $n \to \infty$, for $S_i = \{x \in S : i = \arg\min_j \; d_\phi(x, \mu_j(\infty))\}$, where $\mu_i(\infty))$ is the asymptotically stable equilibrium of (5).

3.1 Case of Constant Local Models

In the special case when locally constant models are used, i.e., when $\hat{f}(x, \theta_i) = \theta_i \in \mathcal{F}$, two-timescale updates are not required, and a simpler solution can be tracked. In particular, we can augment the system (5) with

$$\begin{cases} \sigma_{\theta_i}(n+1) & = \sigma_{\theta_i}(n) + \alpha(n)\left[x_n \hat{p}(\mu_i|x_n) - \sigma_{\theta_i}(n)\right] \\ \theta_i(n) & = \frac{\sigma_{\theta_i}(n)}{\rho_i(n)} \end{cases} \tag{10}$$

Following the same arguments as in the proof of Theorem 1, it is easy to see that $\theta_i(n)$ converge almost surely to $\mathbb{E}\left[\mathbb{1}_{[X\in S_i]}f(X)\right]$ as $n \to \infty$ and $\lambda \to 0$. This approach is equivalent to a piece-wise constant approximation of $f(X)$.

3.2 Classification as Class-Conditioned Density Estimation

As shown in [27], we can formulate the binary classification problem to the minimization of F in (1) with a modified average distortion measure given by

$$d^c(x, c_x, \mu, c_\mu) = \begin{cases} d(x, \mu), \ c_x = c_\mu \\ 0, \ c_x \neq c_\mu \end{cases} \tag{11}$$

It is easy to see that this particular choice for the distortion measure d^c in (11) transforms the learning rule in (5) to

$$\begin{cases} \rho_i(n+1) & = \rho_i(n) + \beta(n) \left[s_i \hat{p}(\mu_i | x_n) - \rho_i(n) \right] \\ \sigma_i(n+1) & = \sigma_i(n) + \beta(n) \left[s_i x_n \hat{p}(\mu_i | x_n) - \sigma_i(n) \right] \end{cases} \tag{12}$$

where $s_i := \mathbb{1}_{\left[c_{\mu_i} = c\right]}$. As a result, this is equivalent to estimating strongly consistent class-conditional density estimators, i.e., $\hat{p}(x|c=j) \to \pi_j p(x|c=j), \ a.s.$

4 Hierarchical Learning in Multiple Resolutions

In this section, we extend the progressive partitioning algorithm (Alg. 1) of Section 2, by imposing a tree structure in the construction of the regions $\{S_i\}$. This structural constraint reduces the time complexity of the algorithm from $O(K^2)$ to $O(k^2 + \log_k K)$, where K here represents the total number of sets $\{S_i\}_{i=1}^K$, and k represents the number of children sub-sets for each parent set (assumed equal for every parent set) [11].

4.1 Tree-Structured Progressive Partitioning

A tree-structured partition $\Sigma_\Delta := \{S_{\nu_i}\}$ is defined by a set of regions $S_{\nu_i} \in S$, each represented by a tree node ν_i, arranged in a tree structure Δ with a single root node ν_0 such that $S_{\nu_0} = S$. The tree structure Δ is a special case of a connected, acyclic directed graph, where each node has a single parent node (except for the root node) and an arbitrary number of children nodes, that is, Δ is not restricted to be a binary tree. The set $C(\nu_i)$ represents the nodes $\{\nu_j\}$ that are children of ν_i, while the set $P(\nu_j)$ represents the node ν_i for which $\nu_j \in C(\nu_i)$. The level $l \geq 0$ of a node $\nu_h \in \Delta$ is the length of the path $\{\nu_0, \ldots, \nu_i, \nu_j, \ldots, \nu_h\}$ leading from the root node ν_0 to ν_h such that $\nu_j \in C(\nu_i)$. The terminal nodes $\tilde{\nu} := \{\nu_i : C(\nu_i) = \emptyset\}$ are called leaves, and the union of their associated sets will be denoted $\tilde{S} := \left\{\tilde{S}_j\right\}$, where $|\tilde{S}| = \tilde{K}$ is the number of leaf sets that create a partition of S, and $\tilde{l} := \max\left\{l : \nu_i^{(l)} \in \tilde{\nu}\right\} < \infty$ will denote the maximum depth of the tree. Σ_Δ defines a hierarchical partitioning scheme for the domain S, such that for every node $\nu_i \in \Delta$ associated with the region S_{ν_i}, its children nodes $\{\nu_j \in C(\nu_i)\}$ are associated with the regions $\left\{S_{\nu_j}\right\}$ that form a partition of S_{ν_i}. We will use the unique paths from the root node as identification label

for each node, i.e., $\nu_j = 0 \dots ij$ such that $\nu_0 = 0$, $C(0) = \{0i\}$, $C(0i) = \{0ij\}$, and so on. As such, Algorithm 1 can be used recursively to construct a tree-structured partition Σ_Δ as follows: Start with node $\nu_0 = 0$ as the only leaf node. Using observations $\{x_n\}$ (realizations of $X \in S$), apply Algorithm 1 until a partition $\{S_{0j}\}$ of $S_0 = S$ is constructed. Then starting with $w = 0$ and for every observation x_n, iterate the process

$$\text{repeat } w \leftarrow w' \in C(w) \text{ such that } x_n \in S_{w'}, \text{ until } C(w) = \emptyset, \tag{13}$$

and apply one stochastic approximation update of Algorithm 1 in S_w. This asynchronous process can continue until the convergence of all applications of Alg. 1, when a finite-depth tree-structured partition Σ_Δ is constructed such that for every node $w \in \Delta$ with children nodes $\{wj\} \in C(w)$, the regions $\{S_{wj}\}$ form a partition of S_w. Training local learning models on Δ using, e.g., the updates in (9) results in a learning structure as illustrated in Fig. 1c.

The time complexity of the tree-structured algorithm is significantly reduced. Let K_{max} be the total number of codevectors allowed. Then Alg. 1 has a worst-case complexity $O(N_c(2\bar{K})^2 d)$, for training, where $\bar{K} = \sum_{n=0}^{\log_2 K_{max}} 2^n$, while testing requires $O(K_{max} d)$ (see Section 2.1 for details on the parameters). In a tree-structured partition Σ_Δ of depth $\tilde{l}$, assuming that the number of children $k = |C(\nu_i)|$ of each node ν_i is k is the same, and that each region is represented by roughly the same number of observations $N_c^{\tilde{l}}$, we get $k = (K_{max})^{1/\tilde{l}}$, and $N_c^{\tilde{l}} = {}^{N_c}/_k$. Then training requires in the worst-case:

$$O\left(\frac{k^{\tilde{l}} - 1}{k(k-1)} N_c (2\bar{k})^2 d)\right)$$

where $\bar{k} = \sum_{n=0}^{\log_2 k} 2^n = \sum_{n=0}^{1/\tilde{l} \log_2 K_{max}} 2^n$. Prediction requires a forward pass of the tree, i.e., it scales with $O(k \log_k K_{max} d)$. In addition, we note that (13) updates the partition $\{S_{wj}\}$ of each node w asynchronously. As a result, a variable-rate coding scheme is constructed that depends on the underlying probability density of the random variable $X \in S$, and often outperforms fixed-rate, full-search techniques with the same average number of bits per sample.

4.2 Multi-Resolution Extension

So far we have modeled the observations as realizations of a random variable $X \in S \subseteq \mathbb{R}^d$. In general, X can be itself a measurable signal $X(t) : \mathbb{R}^n \to S$ with finite energy, i.e., $X(t) \in S \subseteq L^2(\mathbb{R}^d)$. We will denote the original space S as S^0. A multi-resolution representation of the signal $X(t)$ consists of a sequence of projections of $X(t)$ on subspaces $\{S^j\}$ such that $S^j \subset S^{j-1}$, $\forall j \in \mathbb{N}$, and $\cup_{j=0}^{\infty} S^j$ is dense in S^0 with $\cap_{j=0}^{\infty} S^j = \{0\}$. There are numerous methods to construct subspaces $\{S^j\}$ with these properties, from the classical wavelet transform [19] to different dictionary learning approaches [10, 16]. An approach using group-convolutional wavelet decomposition is discussed in Section 4.3.

Algorithm 2 Multi-Resolution Progressive Partitioning

Initialize root node ν_0 s.t. $S_{\nu_0} = S^{\tilde{l}}$
repeat
  Observe data point $x^{\tilde{l}} \in S^{\tilde{l}}$
  Find leaf node to update:
  Set $w = \nu_0$
  Set resolution $l = \tilde{l}$
  while $C(w) \neq \emptyset$ **do**
    $w \leftarrow v \in C(w)$ such that $x^l \in S_v^l$
    $l \leftarrow l - 1$
  end while
  Update partition $\left\{S_{wj}^l\right\}$ of S_w^l using x and Alg. 1
  if Alg. 1 in S_w^l terminates and $l > 0$ **then**
    Split node w: $C(w) \leftarrow \left\{S_{wj}^l\right\}$
  end if
until Stopping criterion

We denote by $X^r \in S^r$ the projection of $X = X^0$ to the subspace S^r. Given a multi-resolution representation of X with subspaces $\left\{S^0, S^1, S^2, \ldots, S^{\tilde{l}}\right\}$, we can extend (13) presented in Section 4.1 such that $X^{\tilde{l}-r} \in S^{\tilde{l}-r}$ is used to train the nodes of the tree at level r. This idea matches the intuition of using higher-resolution representation of X for deeper layers of the tree, and the algorithmic implementation is given in Alg. 2.

4.3 Building Group-Invariant Multi-Resolution Representations

There are numerous methods to construct multi-resolution subspaces $\left\{S^j\right\}$ with the properties mentioned in Section 4.2. In this section we briefly mention a particular approach based on group-convolutional wavelet decomposition that aligns with the principles of the scattering transform, first introduced in [9].

We start with the standard wavelet transform $\{W_l X(t)\}_l \in S^l$ of a signal $X(t) \in S \subseteq L^2(\mathbb{R}^d)$ [19]. The computation of the multi-resolution wavelet representation of a signal consists of successive operations of a linear convolution operator, followed by a downsampling step [21]. As a result, the wavelet transform, is stable to small deformations [9]. In addition, the wavelet transform is translation covariant, that is it commutes with the Lie group of operators $\{T_c\}_{c\in\mathbb{R}}$ such that $T_c X(t) = X(t-c)$, i.e., $W_j(T_c X) = T_c W_j(X)$. In fact, the translation covariance property can be generalized to covariance with respect to the action of an arbitrary compact Lie group G by replacing for the group-convolution operation $(f * h)(x) = \int_G f(g)h(g^{-1}x)dr$, where dr is the Haar measure of G [20].

To induce local invariance (up to a scale 2^J for some $J > 0$) with respect to G, it is sufficient to cascade the wavelet transform with a non-linear operation $\rho W_j X = \|W_j X\|_1$, and a locally averaging integral operation which can be modeled as a convolution with a low-pass filter localized in a spatial window

scaled at 2^J [9]. This is called a scattering transform and its implementation is based on a complex-valued convolutional neural network whose filters are fixed wavelets and ρ is a complex modulus operator as described above [1]. This structure is similar to deep convolutional neural networks [17, 22], where successive operations of a linear convolutional operator, a nonlinear mapping (often a rectifying function, e.g., ReLu), and a down-sampling step (e.g., max-pooling), are used to produce the input for the next stage of the architecture [2, 9, 21].

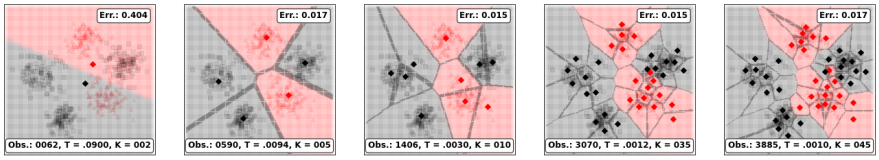
5 Experimental Evaluation and Discussion

We illustrate the properties and evaluate the performance of the proposed learning algorithm in 2D classification problem with class-conditional distributions given by a mixture of 2D Gaussians[3]. The evolution of Alg. 1 is shown in Fig. 2. The temperature level (we use T instead of λ to stress the connection to the temperature level in annealing optimization), 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. This process showcases the bifurcation phenomenon and the performance-complexity trade-off described in Section 2.

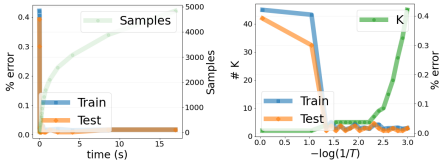
In Fig. 3, we illustrate the evolution of the tree-structured approach. There are two notable comments on the behavior of this approach compared to the original. First, the time complexity is considerably improved (see Section 4), and this results in a drastic difference in the running time of the learning algorithm. In particular, the tree-structured approach achieves a testing error of $e = 99.8\%$ in $t = 0.4s$, while the original approach achieves a testing error of $e = 99.8\%$ in $t = 10.4s$. Secondly, the number of codevectors used is drastically reduced, and the codevectors tend to exist in the boundaries of the Bayes decision surface, instead of populating areas where the decision surface does not fluctuate much.

Finally, the effect of using multiple resolutions as described in Section 4.2, is depicted in Fig. 4 for the same problem as in Fig. 2 and 3. For better visualization, we assume that the low-resolution features, with respect to which the first layer of the tree is computed, are the projections of the two-dimensional data in an one-dimensional space (line). The second layer of the tree is trained using the high-resolution features, i.e., the full knowledge of both coordinates of the data. Notice that this process will converge to a consistent learning algorithm, as long as the multi-resolution representation used complies with the properties mentioned in Section 4.2, and the last layer of the tree uses the full information of the input data.

[3] The open-source code is publicly available at `https://github.com/MavridisChristos/OnlineDeterministicAnnealing`.

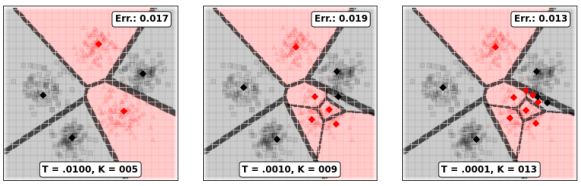


(a) Evolution of the algorithm in the data space.

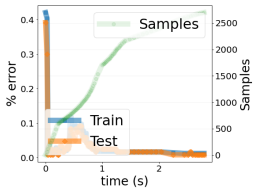


(b) Performance curves.

Fig. 2: Performance curves and data space evolution of the proposed algorithm applied to a classification problem with underlying Gaussian distributions.



(a) Evolution of the algorithm in the data space.

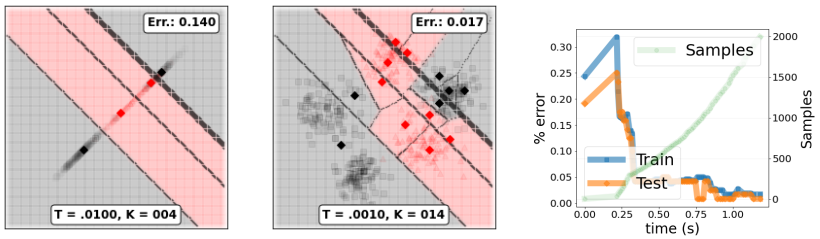


(b) Performance curves.

Fig. 3: Performance curves and data space evolution of the proposed tree-structured algorithm (three layers) applied to a classification problem with underlying Gaussian distributions.

5.1 Tree-Structured Partition, Localization, and Explainability in Machine Learning

We emphasize the advantage of using a tree-structured learning module in the localization properties which allow for an understanding of the input space, in accordance to the principles of the recently introduced class of explainable learning models [24, 31]. The Voronoi regions shrink geometrically, and allow for the use of local models, which is especially important in high-dimensional spaces. Unlike most learning models, it is possible to locate the area of the data space that presents the highest error rate and selectively split it by using local applications of Alg. 1 (ODA). This process can be iterated until the desired error rate (or average distortion) is achieved. This is similar to an often over-fitted classification and regression tree (CART) [8]. However, over-fitting on the training dataset often adversely affects the generalization properties of the model, the performance on the testing dataset, and the robustness against adversarial attacks. Therefore, the progressive process of ODA becomes important in establishing a



(a) Convergence of first layer with low-resolution features.

(b) Convergence of second layer with high-resolution features.

(c) Performance curves.

Fig. 4: Performance curves and data space evolution of the proposed multi-resolution algorithm (two layers) applied to a classification problem with underlying Gaussian distributions.

robust way to control the trade-off between performance and complexity, before you reach that limit. Finally, an important question in tree-structured learning models is the question of which cell to split next. An exhaustive search in the entire tree to find the node that presents the largest error rate is possible but is often not desired due to the large computational overhead. This is automatically circumvented by the proposed multi-resolution ODA approach (Alg. 2) as it asynchronously updates all cells depending on the sequence of the online observations. As a result, the regions of the data space that are more densely populated with data samples are trained first, which results in a higher percentage of performance increase per cell split. We stress that this property makes the proposed algorithm completely dataset-agnostic, in the sense that it does not require the knowledge of a training dataset a priori. Instead operates completely online, i.e., using one observation at a time to update its knowledge base (partition structure and local learning models) [23, 28, 30].

6 Conclusion

We introduced a hierarchical learning algorithm based on the principles of online deterministic annealing. The learning architecture simulates an annealing process and defines a heuristic method to progressively construct a tree-structured partition of a possibly multi-resolution data space, which can be used in conjunction with general learning algorithms to train local models. The structured partitioning of the input space provides explainability, and makes the learning architecture a suitable candidate for transfer learning applications. Finally, the use of online gradient-free training rule based on stochastic approximation updates allows for the use of the proposed method for inference, control, and reinforcement learning applications.

References

1. Andreux, M., Angles, T., Exarchakis, G., Leonarduzzi, R., Rochette, G., Thiry, L., Zarka, J., Mallat, S., Andén, J., Belilovsky, E., Bruna, J., Lostanlen, V., Hirn, M.J., Oyallon, E., Zhang, S., Cella, C., Eickenberg, M.: Kymatio: Scattering transforms in python (2019)
2. Anselmi, F., Rosasco, L., Tan, C., Poggio, T.: Deep convolutional networks are hierarchical kernel machines. arXiv preprint arXiv:1508.01084 (2015)
3. Banerjee, A., Merugu, S., Dhillon, I.S., Ghosh, J.: Clustering with bregman divergences. Journal of machine learning research **6**(Oct), 1705–1749 (2005)
4. Bennett, K.P., Parrado-Hernández, E.: The interplay of optimization and machine learning research. The Journal of Machine Learning Research **7**, 1265–1281 (2006)
5. Biehl, M., Hammer, B., Villmann, T.: Prototype-based models in machine learning. Wiley Interdisciplinary Reviews: Cognitive Science **7**(2), 92–111 (2016)
6. Borkar, V.S.: Stochastic approximation with two time scales. Systems & Control Letters **29**(5), 291–294 (1997)
7. Borkar, V.S.: Stochastic approximation: a dynamical systems viewpoint, vol. 48. Springer (2009)
8. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
9. Bruna, J., Mallat, S.: Invariant scattering convolution networks. IEEE transactions on pattern analysis and machine intelligence **35**(8), 1872–1886 (2013)
10. Chen, S.S., Donoho, D.L., Saunders, M.A.: Atomic decomposition by basis pursuit. SIAM review **43**(1), 129–159 (2001)
11. Gray, R.M.: Vector quantization. Readings in speech recognition **1**(2), 75–100 (1990)
12. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural computation **18**(7), 1527–1554 (2006)
13. Hüllermeier, E., Waegeman, W.: Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. Machine Learning **110**, 457–506 (2021)
14. Kohonen, T.: Learning Vector Quantization, pp. 175–189. Springer Berlin Heidelberg, Berlin, Heidelberg (1995)
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
16. LeCun, Y.: The next frontier in ai: Unsupervised learning. https://www.youtube.com/watch?v=IbjF5VjniVE (2016)
17. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)
18. Lee, H., Grosse, R., Ranganath, R., Ng, A.Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: Proceedings of the 26th annual international conference on machine learning. pp. 609–616 (2009)
19. Mallat, S.: A wavelet tour of signal processing. Elsevier (1999)
20. Mallat, S.: Group invariant scattering. Communications on Pure and Applied Mathematics **65**(10), 1331–1398 (2012)
21. Mallat, S.: Understanding deep convolutional networks. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **374**(2065), 20150203 (2016)
22. Mavridis, C., Baras, J.: Multi-resolution online deterministic annealing: A hierarchical and progressive learning architecture. arXiv preprint arXiv:2212.08189 (2022)

23. Mavridis, C., Baras, J.S.: Annealing optimization for progressive learning with stochastic approximation. IEEE Transactions on Automatic Control **68**(5), 2862–2874 (2023)
24. Mavridis, C., Noorani, E., Baras, J.S.: Risk sensitivity and entropy regularization in prototype-based learning. In: 2022 30th Mediterranean Conference on Control and Automation (MED). pp. 194–199. IEEE (2022)
25. Mavridis, C.N., Baras, J.S.: Convergence of stochastic vector quantization and learning vector quantization with bregman divergences. IFAC-PapersOnLine **53**(2) (2020)
26. Mavridis, C.N., Baras, J.S.: Progressive graph partitioning based on information diffusion. In: IEEE Conference on Decision and Control. pp. 37–42 (2021)
27. Mavridis, C.N., Baras, J.S.: Online deterministic annealing for classification and clustering. IEEE Transactions on Neural Networks and Learning Systems **34**(10), 7125–7134 (2023). https://doi.org/10.1109/TNNLS.2021.3138676
28. Mavridis, C.N., Kanellopoulos, A., Vamvoudakis, K., Baras, J.S., Johansson, K.H.: Attack identification for cyber-physical security in dynamic games under cognitive hierarchy. IFAC-PapersOnLine (2023)
29. Mavridis, C.N., Kontoudis, G.P., Baras, J.S.: Sparse gaussian process regression using progressively growing learning representations. In: 2022 IEEE 61st Conference on Decision and Control (CDC). pp. 1454–1459. IEEE (2022)
30. Mavridis, C.N., Suriyarachchi, N., Baras, J.S.: Detection of dynamically changing leaders in complex swarms from observed dynamic data. In: International Conference on Decision and Game Theory for Security. pp. 223–240. Springer (2020)
31. Milani, S., Topin, N., Veloso, M., Fang, F.: A survey of explainable reinforcement learning. arXiv preprint arXiv:2202.08434 (2022)
32. Rüping, S.: Learning with local models. In: Morik, K., Boulicaut, J.F., Siebes, A. (eds.) Local Pattern Detection. pp. 153–170. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
33. Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for deep learning in nlp. arXiv preprint arXiv:1906.02243 (2019)
34. Thompson, N.C., Greenewald, K., Lee, K., Manso, G.F.: The computational limits of deep learning. arXiv preprint arXiv:2007.05558 (2020)