HebbAL: A Hebbian-Based Strategy for Kernel Active Learning

Yan V. G. Ferreira
Department of Electronic Engineering
Universidade Federal de Minas Gerais, UFMG
Av. Antônio Carlos 6627, 31270-901
Belo Horizonte, MG, Brazil
yanvgferreira@gmail.com

Elias J. R. Freitas
Graduate Program in Electrical Engineering
Universidade Federal de Minas Gerais, UFMG
Av. Antônio Carlos 6627, 31270-901
Belo Horizonte, MG, Brazil
Federal Institute of Minas Gerais, IFMG
elias.freitas@ifmg.edu.br

Luiz C. B. Torres
Department of Computing and Systems
Universidade Federal de Ouro Preto
João Monlevade, Brazil
luiz.torres@ufop.edu.br

Antonio P. Braga
Department of Electronic Engineering
Universidade Federal de Minas Gerais, UFMG
Av. Antônio Carlos 6627, 31270-901
Belo Horizonte, MG, Brazil
apbraga@ufmg.br

***Abstract*—Although most machine learning methods rely on the availability of large sets of labeled training instances, most real-world available data is unlabeled, since labeling can be expensive and time-consuming. To avoid the expense of labeling all the data before fitting a model, the paradigm of active learning allows models to query the labeling of only the most relevant samples to its training. However, when active learning is performed online, specially with streaming data, updating the model can be challenging, as it usually requires running several optimization steps or inverting matrices to retrain a model. In fact, this can make learning unfeasible for systems with computational constraints, such as embedded hardware. In this context, the present work aims at enhancing active learning with Hebbian classifiers, which allow updating parameters inexpensively to learn new samples or to remove outdated information. For this purpose, we employ a recently proposed kernel-based method for embedding data in a suitable way for Hebbian learning and combine it with a querying criterion based on the Perceptron Convergence Theorem. The resulting model, called HebbAL, is applied to 17 UCI benchmarks, presenting data to the model in a streaming fashion. Our approach achieved comparable performance to widely used active learning models, such as a Random Forest with uncertainty querying. Although HebbAL presents a slightly lower AUC on average, it queries less labels than the other models. Therefore, our novel approach to online active learning represents a step towards building scalable learning systems to act in dynamical environments.**

***Index Terms*—Active learning, Hebbian learning, kernel learning, online learning.**

I. Introduction

Supervised learning relies on the assumption that representative labeled datasets are available in order to induce a general approximation function. However, in practice, most available data is unlabeled, since assigning classes to samples usually demands the judgment of a human – in some cases, an expert –, which can make this process slow and expensive [1].

To avoid the expense of labeling all the data, the active learning paradigm [2] was proposed to allow machine learning models themselves automatically select only the most significant samples to be labeled. This is commonly achieved measuring either a sample's novelty relative to labeled instances or the model's uncertainty about the classification of the pattern [3]. In particular, uncertainty sampling gained popularity recently for a large range of applications as a querying strategy. For instance, Shankaranarayana [4] recently trained tree-based models on tabular data using this strategy, achieving superior performance in relation to other popular strategies both in classification and regression tasks.

When learning in dynamic environments, a model should also be able to update its knowledge base as the behavior of the system changes over time. In robotics, for instance, when exploring an unknown environment, a robot should collect the most relevant information about it and adapt itself to cope with the new circumstances. Therefore, a special paradigm of active learning, called online active learning [5], emerged as a way to comprise, beyond the selection of the most relevant training samples, real-time model adaptation.

However, most online learning algorithms demand solving time-consuming optimization problems, often involving several matrix multiplications or inversions. For larger models, this can become unfeasible in time or compute-constrained systems, such as embedded hardware. To over-

This work was carried out with the support of the Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – Campus Ibirité. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

come this limitation, a simpler training paradigm, known as Hebbian learning [6, 7], may be preferred over the traditional options. Based solely on dot products between input and output data, without explicit error minimization, this paradigm allows updating model weights inexpensively. Nevertheless, Hebbian learning demands linearly separable and mutually orthogonal data to approach the least squares solution, which demands a special treatment on the input data prior to training.

In this sense, this work proposes an approach to make the Hebbian training paradigm feasible in online active learning. Our method, named HebbAL, is based on a recently proposed embedding technique which uses MLP kernels [8, 9] to project data in a 2-dimensional likelihood space [6]. In this space, data become linearly separable and partially orthogonal, allowing the training of a Hebbian classifier. We then combine this embedding with an active Hebbian learning technique that queries labels for incoming data based on the Perceptron Convergence Theorem [10]. Our approach is compared to other online active learning techniques on 17 UCI benchmarks.

The remainder of the work is organized as follows. Section II presents a brief review of online active learning and describes Hebbian active learning. Section III describes the kernel-based projection method which our method is based on. Section IV details the proposed HebbAL approach. Experimental results and their analysis are presented in Section V. Finally, Section VI summarizes the main contributions and outlines potential directions for future research.

II. Online Active Learning

Online active learning is a machine learning paradigm in which new unlabeled data – possibly comprising new behaviors of the modeled system – become available over time and an active learner, a model capable of querying the labels of only the samples it identifies as the most relevant ones, is continuously updated. This approach is particularly useful in dynamic nonstationary environments, where new behaviors may arise from changes in operational conditions or aging of sensors and actuators, for instance, and labeling is expensive (e.g., an industrial plant whose variables are measured through a slow chemical laboratory process).

Generally speaking, more labeled instances yield higher model accuracy. However, active learning methods can achieve similar accuracy with fewer labeled data through intelligent instance selection. Fig. 1 illustrates the learning curve of the same model (a Random Forest) trained with and without active learning in a simple synthetic problem, highlighting the efficiency of active learning in reducing the number of required labels while maintaining or improving model performance.

Algorithm 1 outlines a general online active learning algorithm. Initially, a model is built from a reduced set of labeled instances. Then, the algorithm intelligently selects and adds only significant instances to the training data. The model is iteratively updated until a stopping criterion, such as achieving a certain accuracy, is met.

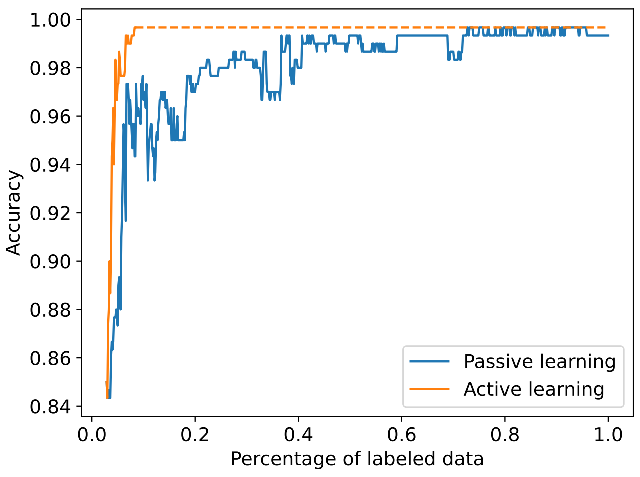

Fig. 1. Evolution of the accuracy of the same model with an active and a passive learning strategy as more labeled samples are used from the same dataset. Since active learning queries the labeling of only the most significative samples, it achieves greater performance, requiring less labeled data than passive learning.

Algorithm 1: General Online Active Learning

```
1  Input: Instances
2  Output: Trained model
3  Fit a model with initial labeled instances;
4  while not termination criteria do
5      Select an unlabeled instance from the pool;
       /* Intelligent selection (query)                */
6      if Is it a Significant Instance? then
7          Obtain the label of this selected instance;
8          Append it to the training data;
9          Update the model;
10     end
11 end
```

By utilizing fewer labeled instances, active learning methods offer an efficient way to maintain or enhance model accuracy, making them particularly valuable for applications where labeling is costly or time-consuming.

A. Hebbian Active Learning

A major challenge in learning from dynamic environments is adapting a model's parameters as new data becomes available. Most online learning methods address this by performing multiple iterations of an optimization algorithm, which can be computationally intensive and impractical for embedded systems with limited resources. In such scenarios, Hebbian learning offers a promising alternative: instead of relying on iterative optimization, it updates the model using inexpensive additive adjustments to the parameters whenever a new sample is learned or an old one is discarded, requiring no matrix inversion or gradient descent steps.

Hebb's Rule [7], initially proposed as a model for synaptic adaptation in biological neurons, is often summarized by the phrase "neurons that fire together, wire together." Due to its simplicity and biological plausibility, Hebbian learning has long been employed in training artificial neural networks [11]. For a single linear neuron with p inputs, the weights $\boldsymbol{w}$ can be computed directly from the data using:

$$\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{Y}, \tag{1}$$

where $\boldsymbol{X}$ is an $N \times p$ input matrix consisting of N samples and $\boldsymbol{Y}$ is the corresponding N-dimensional target vector. In this formulation, each weight w_i is updated based on the correlation between the i-th input feature vector $\boldsymbol{p}_i$ and the target values.

This update mechanism allows for efficient online learning. When a new labeled sample $(\boldsymbol{x}_{\text{new}}, y_{\text{new}})$ arrives, the weights are updated by simply adding the product $\boldsymbol{x}_{\text{new}}^T y_{\text{new}}$ to the current weights. Conversely, to "forget" a sample $(\boldsymbol{x}_{\text{old}}, y_{\text{old}})$, one subtracts the term $\boldsymbol{x}_{\text{old}}^T y_{\text{old}}$. These operations can be performed inexpensively in compute or time-constrained embedded systems.

However, Equation 1 provides an optimal solution only when the input samples are mutually orthogonal. In general, non-orthogonality introduces an error term known as crosstalk, which degrades the model's performance. As demonstrated by Ushikoshi *et al.* [6], the prediction for a normalized training sample $\boldsymbol{x}_i$ becomes:

$$\hat{y}_i = \sum_{j=1}^{N} \boldsymbol{x}_i \cdot \boldsymbol{x}_j y_j \tag{2}$$

$$= y_i + \sum_{\substack{j=1 \\ j\neq i}}^{N} \boldsymbol{x}_i \cdot \boldsymbol{x}_j y_j, \tag{3}$$

where the second term reflects interference from non-orthogonal samples. When the inputs are orthogonal, the Hebbian solution coincides with the least-squares solution, and crosstalk vanishes. Nonetheless, since Hebb's rule describes the training of a single linear neuron, the approach is inherently limited to linearly separable problems.

To extend Hebbian learning to nonlinear settings and streaming data, Horta *et al.* [10] proposed combining it with the framework of Extreme Learning Machines (ELMs) [12], a class of Single Layer Feedforward Networks (SLFNs). In this architecture, input data $\boldsymbol{X}$ are projected into a high-dimensional feature space via random weights $\boldsymbol{Z}$, followed by a nonlinear activation function Φ – usually a ReLU or a hyperbolic tangent –, yielding the hidden-layer output matrix $\boldsymbol{H}$:

$$\boldsymbol{H} = \Phi(\boldsymbol{X}\boldsymbol{Z}). \tag{4}$$

According to Cover's Theorem [13], projecting data into a sufficiently high-dimensional nonlinear feature space increases the likelihood that it becomes linearly separable. Within this transformed space, the output layer can be trained using Hebbian learning. Horta *et al.* employed a normalized variant of Hebb's rule [14], training a linear model in feature space as

$$\boldsymbol{w} = \frac{\boldsymbol{H}^T\boldsymbol{Y}}{\|\boldsymbol{H}^T\boldsymbol{Y}\|}. \tag{5}$$

Beyond enabling fast and efficient parameter updates in data streams, the authors also observed that the residual crosstalk error has a regularization effect on the solution, helping to prevent overfitting in the high-dimensional space.

To enable active learning in this setting, Horta *et al.* proposed a novel query strategy grounded in the Perceptron Convergence Theorem. The model queries the label for an unlabeled instance $\boldsymbol{x}_i \in \xi$ (where ξ denotes the unlabeled pool) whenever the current number of labeled samples m is insufficient to guarantee convergence:

$$m < \frac{\beta_i + 2\theta_i}{\alpha_i^2}, \tag{6}$$

where the parameters α_i, β_i, and θ_i are updated iteratively as follows:

$$\begin{aligned} \alpha_i &= \left|\boldsymbol{w}^T\boldsymbol{x}_i\right|, \\ \beta_i &= \max\left(\|\boldsymbol{x}_i\|^2, \beta_{i-1}\right), \\ \theta_i &= \max\left(\left|\boldsymbol{w}^T\boldsymbol{x}_i\right|, \theta_{i-1}\right). \end{aligned} \tag{7}$$

This criterion prioritizes the selection of instances that most contribute to ensuring the convergence of the Hebbian perceptron in the high-dimensional feature space, enabling effective learning with a minimal number of labels.

III. Kernel-based Hebbian Learning

Recently, Ushikoshi *et al.* [6] proposed a Hebbian classifier that uses kernel embeddings to linearize data while minimizing crosstalk, achieving remarkable performance in passive learning tasks. Unlike the method by Horta *et al.* [10], linearization and orthogonalization within this framework are achieved by learning a kernel that projects the data into a two-dimensional space of class likelihoods – eliminating the need for high-dimensional mappings, which can lead to overfitting.

A kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is defined as the inner product of two input vectors (x_i and x_j) mapped by a function $\rho(.)$,

$$K(x_i, x_j) = \langle \rho(x_i), \rho(x_j) \rangle, \tag{8}$$

and can be interpreted as a similarity measure between samples. For instance, a popular kernel function is the Gaussian kernel, $K(x_i, x_j) = e^{-\gamma\|x_i - x_j\|^2}$. More generally, universal approximators, such as neural networks, can be employed as kernels, allowing to learn the best representation for each problem. In fact, Ushikoshi *et al.* adopt the MLP kernel embedding proposed by Menezes *et al.* [8, 9] to project training samples into a likelihood space, which allows better separation between classes, as illustrated in Figure 2.

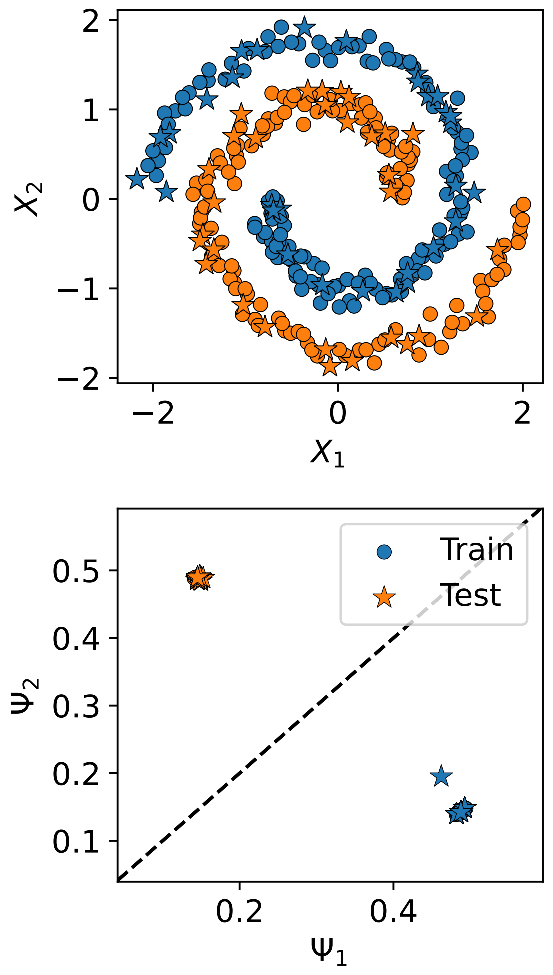

Fig. 2. Projection of data in likelihood space using MLP kernel.

The likelihood – or, in the general case, the similarity – of a sample $\boldsymbol{x}_j$ in relation to a class C_i is defined as

$$\Psi(\boldsymbol{x}_j, C_i) = \frac{1}{N_{C_i}} \sum_{k=1}^{N_{C_i}} K(\boldsymbol{x}_j, \boldsymbol{x}_k), \tag{9}$$

where N_{C_i} is the number of samples of class C_i. Therefore, in a binary classification problem, two similarities are associated to each input sample – one for each class – and the data can be projected into a 2D space $\Psi_1 \times \Psi_2$ of those similarities.

Menezes *et al.* further extend this idea to define the similarity between two classes C_i and C_j with N_{C_i} and N_{C_j} samples, respectively, as the mean similarity of the samples from C_i with respect to class C_j:

$$\mathrm{S}_{i,j}(C_i, C_j) = \frac{1}{N_{C_i}} \sum_{k=1}^{N_{C_i}} \Psi(\boldsymbol{x}_k, C_j). \tag{10}$$

This leads to a matrix $\boldsymbol{V}$ of within-class and between-class similarities:

$$\boldsymbol{V} = \begin{bmatrix} S_{ii} & S_{ij} \\ S_{ji} & S_{jj} \end{bmatrix} = \begin{bmatrix} \boldsymbol{V}_i^\top \\ \boldsymbol{V}_j^\top \end{bmatrix}, \tag{11}$$

where $\boldsymbol{V}_i = [S_{ii}\ S_{jj}]^\top$ and $\boldsymbol{V}_j = [S_{ji}\ S_{jj}]^\top$ are exactly the class centroids in the similarity space.

The likelihood values depend on the kernel function parameters $\boldsymbol{\Theta}$ – for MLP kernels, the synaptic weights of the network. A measure of separability can then be defined in terms of the chosen kernel parameters $\boldsymbol{\Theta}$, interpreted as a quality function from $\mathbb{R}^2$ to $\mathbb{R}$ in binary classification contexts. Menezes *et al.* define this quality function as the distance between the vectors $\boldsymbol{V}_i$ and $\boldsymbol{V}_j$,

$$\mathcal{D}(X, \boldsymbol{\Theta}) = \|\boldsymbol{V}_1 - \boldsymbol{V}_2\|. \tag{12}$$

The optimal kernel with parameters $\boldsymbol{\Theta}^*$ maximizes the distance between the clusters of each class in likelihood space, leading to linear separability. Therefore, they train the MLP kernel with gradient descent to solve it:

$$\boldsymbol{\Theta}^* = \arg\max_{\Theta} \mathcal{D}(\mathrm{X}, \Theta), \tag{13}$$

Beyond linearizing the data, Ushikoshi *et al.* show that this embedding is particularly suitable for Hebbian learning, since it minimizes inter-class crosstalk. To fully orthogonalize a dataset composed of N samples, leading the crosstalk term to zero, at least $p = N$ input variables would be needed, since the maximum number of linearly independent vectors in $\mathbb{R}^p$ is p. In this case, however, models would likely overfit the data, since the number of training samples would be equal or less than the number of free parameters. Therefore, if crosstalk is not too large, it may actually be desirable, acting as a regularizer for the models [10]. The MLP kernel embedding helps balancing this trade-off by minimizing only the crosstalk between samples of opposite classes, which the authors call inter-class crosstalk. For a binary classification problem, this term is defined as

$$I_{i,j} = \frac{1}{N_{C_i}} \sum_{l=1}^{N_{C_i}} \sum_{k=1, k\neq l}^{N_{C_j}} \boldsymbol{x}_l \boldsymbol{x}_k^T y_k \tag{14}$$

and is minimized as the two classes approach orthogonality in the likelihood space.

Ushikoshi *et al.* report that the Hebbian Perceptron trained in likelihood space achieved comparable performance to the state-of-the-art across several datasets, suggesting that kernel-based embeddings of data can enhance the effectiveness of Hebbian learning. Although they mention that the model could be extended to online learning due to its fast adaptation capabilities, the authors limit their experiments to offline and passive learning tasks.

IV. Methods

The feasibility of Hebbian active learning in streaming scenarios strongly depends on effective data linearization and minimizing crosstalk without causing the model to overfit. While the method by Horta *et al.* achieves linearization and implements an effective querying criterion, it lacks mechanisms for controlling crosstalk. And, since crosstalk increases with the number of non-orthogonal training samples, this could lead to performance degradation in scenarios where large amounts of data are fed to the model continuously. In contrast, Ushikoshi *et al.* describe a mechanism for data linearization in a likelihood space which, additionally, helps minimizing crosstalk between samples of opposite classes. However, the authors do not apply their methodology for online active learning, therefore not taking advantage of the main benefit of Hebb's Rule: fast and lightweight adaptation for streaming data.

In this work, we leverage the strengths of both approaches to propose HebbAL, a Hebbian active learning method that benefits from likelihood space projections. Our methodology is detailed in Algorithm 2. Provided a pool of labeled data $(\boldsymbol{X}_l, \boldsymbol{Y}_l)$ and a pool of unlabeled data $\boldsymbol{X}_u$, the MLP kernel is trained on the labeled data according to Equation 13, and a Hebbian classifier is initialized. Then, each unlabeled sample that arrives is projected into the likelihood space and its significance is tested using Horta's criterion. If the sample is significant, the weight vector of the model is updated. When few initial labeled samples are available, updating the MLP kernel as new labels become available can be beneficial, as projection quality depends on its generalization ability.

Our methodology was applied to 17 binary classification benchmarks from University of California Irvine (UCI), as listed in Table I. We compare HebbAL to Horta's Extreme Active Learning Machine (EALM) [10] and to a Random Forest (RF) classifier using classifier uncertainty as querying method, a very popular approach to active learning in tabular datasets. Since both HebbAL and RF demand an initial pool of labeled samples, we initialized them with 5%, 10%, and 20% of the full training data. For each dataset, a 10-fold stratified cross-validation was performed, with the training samples being presented in a streaming fashion to the models. The area under the ROC curve (AUC) of the final trained models was calculated, and the total number of labeled samples used (initial and selected) was registered.

V. Results and Discussion

The results of our experiments are presented in Tables II and III. Table II presents the average and standard deviation of the AUC scores for each dataset, while Table III reports the number of labeled samples used by each model. Based on the average AUC ranking, the models with the best performance were the Random Forest initialized with 20% of the training data and EALM, which does not require initialization. Although HebbAL achieved competitive results on most datasets, its average performance was lower than the other models, with the model initialized with 5% of the training samples performing the worst overall. This drop in performance can be

Algorithm 2: HebbAL

```
Input: Labeled data: $(\boldsymbol{X}_l, \boldsymbol{Y}_l)$, Unlabeled data: $\boldsymbol{X}_u$
Output: Weight vector: $\boldsymbol{w}$, Space function: $\Psi$
/* Pool of selected samples                                  */
$(\boldsymbol{X}_s, \boldsymbol{Y}_s) \leftarrow (\boldsymbol{X}_l, \boldsymbol{Y}_l)$;
/* Train MLP kernel                                          */
$\Psi = \mathrm{fit}(\boldsymbol{X}_l, \boldsymbol{Y}_l)$;
/* Projection into likelihood space                          */
$\boldsymbol{H}_l = \Psi(\boldsymbol{X}_l)$;
/* Standardization and bias term                             */
$\boldsymbol{\mu} = \mathrm{mean}(\boldsymbol{H}_l)$; $\boldsymbol{\sigma} = \mathrm{std}(\boldsymbol{H}_l)$;
$\boldsymbol{\mathcal{H}_l} = \left[\frac{(\boldsymbol{H}_l - \boldsymbol{\mu})}{\boldsymbol{\sigma}}; \vec{1}\right]$;
/* Normalized Hebbian rule                                   */
$\boldsymbol{w} = \frac{\boldsymbol{\mathcal{H}}_l^T \cdot \boldsymbol{Y_l}}{\left\|\boldsymbol{\mathcal{H}}_l^T \cdot \boldsymbol{Y_l}\right\|}$;
$m$ = Size of $\boldsymbol{X}_l$;
Initialize $\alpha$, $\beta$, $\theta$;
/* Get and check unlabeled samples                           */
for $\boldsymbol{x}_i$ in $\boldsymbol{X}_u$ do
    $\boldsymbol{h}_i = \Psi(\boldsymbol{x}_i)$;
    $\boldsymbol{\hbar_i} = \left[\frac{(\boldsymbol{h_i} - \boldsymbol{\mu})}{\boldsymbol{\sigma}}; \vec{1}\right]$;
    $\alpha = \left|\boldsymbol{w}^T \cdot \boldsymbol{\hbar_i}\right|$;
    $\beta = \max(\left\|\boldsymbol{\hbar_i}\right\|^2, \beta)$;
    $\theta = \max(\left|\boldsymbol{w}^T \cdot \boldsymbol{\hbar_i}\right|, \theta)$;
    $t = \frac{\beta + 2\theta}{\alpha^2}$;
    if $m < t$ then
        Query output $\boldsymbol{y}_i$;
        Add $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ to $(\boldsymbol{X}_s, \boldsymbol{Y}_s)$;
        /* Update kernel (if needed)                         */
        $\Psi = \mathrm{fit}(\boldsymbol{X}_s, \boldsymbol{Y}_s)$;
        $\boldsymbol{H_s} = \Psi(\boldsymbol{X_s})$;
        $\boldsymbol{\mu} = \mathrm{mean}(\boldsymbol{h_s})$; $\boldsymbol{\sigma} = \mathrm{std}(\boldsymbol{h_s})$;
        $\boldsymbol{\mathcal{H}_s} = \left[\frac{(\boldsymbol{H_s} - \boldsymbol{\mu})}{\boldsymbol{\sigma}}; \vec{1}\right]$;
        /* Update weight vector                              */
        $\boldsymbol{w} = \frac{\boldsymbol{\mathcal{H}}_s^T \cdot \boldsymbol{y_s}}{\left\|\boldsymbol{\mathcal{H}}_s^T \cdot \boldsymbol{y_s}\right\|}$;
        $m = m + 1$;
    end
end
```

TABLE I
Evaluated Datasets.

Dataset	# Features	# Samples	Class 1 (%)	Class 2 (%)
australian	14	690	55.51	44.49
breastcancer	12	569	62.74	37.26
bupa	6	345	57.97	42.03
climate	20	540	8.52	91.48
diabetes	8	768	34.9	65.1
german	24	1000	30.00	70.00
haberman	3	306	26.47	73.53
heart	13	270	55.56	44.44
ilpd	10	579	28.50	71.50
ionosphere	33	351	35.90	64.10
parkinsons	22	195	24.62	75.38
sonar	60	208	46.63	53.37
spirals	2	300	50.00	50.00
pima	8	768	34.9	65.1
wisconsin	30	569	62.7	37.3
banknote	4	1372	55.54	44.46
spambase	57	4601	39.4	60.6

TABLE II

Average AUC (Mean ± Standard Deviation) for different Active Learning (AL) strategies during 10-fold cross-validation. Abbreviations denote: EALM (Extreme Active Learning Machine [10]), HebbAL (proposed method with X% initial labels), and RF+U (Random Forest + Classifier Uncertainty with X% initial labels).

Dataset	EALM	HebbAL (5%)	HebbAL (10%)	HebbAL (20%)	RF+U (5%)	RF+U (10%)	RF+U (20%)
australian	92.39 ± 2.65	88.96 ± 4.30	88.99 ± 4.43	90.85 ± 3.98	92.05 ± 3.41	92.14 ± 2.94	93.70 ± 2.86
breastcancer	98.56 ± 1.15	97.97 ± 1.46	97.05 ± 2.11	98.13 ± 1.29	98.62 ± 1.10	98.38 ± 1.29	98.50 ± 1.19
bupa	62.56 ± 10.42	63.37 ± 8.15	65.62 ± 12.37	70.53 ± 4.92	66.76 ± 8.24	68.14 ± 11.26	77.72 ± 5.28
climate	83.28 ± 11.97	89.05 ± 12.13	88.47 ± 10.40	89.50 ± 12.90	72.78 ± 19.08	68.06 ± 13.82	78.36 ± 10.21
diabetes	81.89 ± 5.07	77.23 ± 6.09	74.30 ± 8.16	76.34 ± 4.48	80.27 ± 5.26	81.36 ± 4.50	80.90 ± 4.60
german	78.66 ± 4.62	71.28 ± 6.52	70.84 ± 5.81	72.21 ± 6.44	77.00 ± 5.99	76.32 ± 5.67	77.73 ± 5.25
haberman	66.53 ± 10.10	60.21 ± 8.98	63.28 ± 15.84	64.78 ± 12.18	65.10 ± 8.67	63.71 ± 15.00	64.05 ± 9.54
heart	91.00 ± 4.91	83.39 ± 9.90	84.61 ± 6.04	88.39 ± 3.03	89.69 ± 5.77	90.69 ± 3.24	88.47 ± 4.24
ilpd	57.93 ± 8.23	63.16 ± 8.00	65.11 ± 9.80	71.72 ± 5.01	62.80 ± 10.72	72.67 ± 5.85	70.92 ± 5.22
ionosphere	93.78 ± 6.08	90.38 ± 5.63	90.28 ± 6.87	90.33 ± 4.03	95.22 ± 3.53	96.03 ± 4.04	96.33 ± 3.13
parkinsons	85.74 ± 10.07	84.95 ± 8.64	87.19 ± 8.92	90.97 ± 8.63	89.07 ± 6.32	90.23 ± 9.06	91.59 ± 5.52
sonar	86.80 ± 7.60	80.29 ± 13.00	80.32 ± 11.39	80.77 ± 7.83	89.66 ± 7.01	89.36 ± 10.37	89.41 ± 7.97
spirals	58.13 ± 9.75	65.02 ± 15.66	78.47 ± 9.31	78.93 ± 10.53	90.13 ± 7.85	92.89 ± 8.45	99.04 ± 2.04
pima	81.66 ± 2.48	76.79 ± 5.13	76.62 ± 6.55	79.12 ± 2.96	81.19 ± 3.88	79.52 ± 3.77	81.92 ± 4.29
wisconsin	99.52 ± 0.75	98.24 ± 1.99	99.31 ± 0.92	99.08 ± 1.15	99.42 ± 0.66	99.16 ± 0.82	99.17 ± 0.68
banknote	100.00 ± 0.01	99.80 ± 0.17	99.94 ± 0.16	99.91 ± 0.19	99.84 ± 0.26	99.82 ± 0.46	99.92 ± 0.13
spambase	96.22 ± 0.73	94.73 ± 0.97	95.01 ± 0.93	95.66 ± 1.08	98.00 ± 0.63	98.12 ± 0.48	98.17 ± 0.56
Average ranking	3.23	6.05	5.41	4.18	3.35	3.41	2.35

TABLE III

Average number of queried labels (Mean ± Standard Deviation) for different Active Learning (AL) strategies during 10-fold cross-validation. Abbreviations denote: EALM (Extreme Active Learning Machine [10]), HebbAL (proposed method with X% initial labels), and RF+U (Random Forest + Classifier Uncertainty with X% initial labels).

Dataset	EALM	HebbAL (5%)	HebbAL (10%)	HebbAL (20%)	RF+U (5%)	RF+U (10%)	RF+U (20%)
australian	117 ± 9	47 ± 7	70 ± 3	128 ± 2	108 ± 10	125 ± 7	180 ± 13
breastcancer	62 ± 3	30 ± 3	54 ± 2	104 ± 2	48 ± 4	67 ± 5	116 ± 4
bupa	109 ± 21	41 ± 13	54 ± 7	75 ± 3	90 ± 27	106 ± 27	137 ± 9
climate	108 ± 22	103 ± 26	111 ± 29	142 ± 24	28 ± 9	48 ± 1	98 ± 1
diabetes	170 ± 8	62 ± 10	91 ± 5	150 ± 4	163 ± 18	191 ± 13	248 ± 11
german	305 ± 9	101 ± 16	124 ± 10	199 ± 5	250 ± 35	284 ± 32	346 ± 18
haberman	49 ± 9	59 ± 25	62 ± 22	67 ± 15	40 ± 12	52 ± 10	87 ± 7
heart	74 ± 3	25 ± 6	34 ± 4	52 ± 2	63 ± 6	72 ± 9	86 ± 6
ilpd	101 ± 25	77 ± 17	78 ± 5	122 ± 7	89 ± 39	142 ± 18	189 ± 9
ionosphere	102 ± 5	30 ± 6	44 ± 8	70 ± 3	54 ± 6	60 ± 4	82 ± 3
parkinsons	71 ± 7	38 ± 8	41 ± 12	53 ± 10	28 ± 10	30 ± 8	49 ± 6
sonar	99 ± 5	32 ± 7	35 ± 10	44 ± 3	76 ± 6	82 ± 11	89 ± 8
spirals	65 ± 19	46 ± 11	53 ± 12	68 ± 9	38 ± 7	46 ± 6	69 ± 2
pima	174 ± 5	59 ± 6	90 ± 7	150 ± 3	175 ± 26	190 ± 24	248 ± 12
wisconsin	76 ± 3	31 ± 4	54 ± 2	104 ± 2	55 ± 5	77 ± 4	122 ± 2
banknote	151 ± 4	64 ± 1	124 ± 1	248 ± 1	88 ± 6	143 ± 3	259 ± 2
spambase	557 ± 14	227 ± 4	425 ± 4	833 ± 2	476 ± 12	647 ± 22	1005 ± 14
Average ranking	5	1.55	2.76	4.88	2.82	4.44	6.53

explained considering the average ranking of the number of queried labels on Table III. Except for HebbAL with 20% initial labels, our approach used the fewest samples, and the models with the highest AUC used the most. Although active learning aims to use the smallest training set possible to learn a task, our experiments suggest that this came at the expense of accuracy.

To analyze how each model balances accuracy and data usage across the datasets, we combined both metrics in Fig. 3. Even though our approaches consistently queried less than 30% of the data – and often stayed below 20% –, very high AUC was achieved in many datasets. Remarkably, HebbAL with 5% or 10% initialization had an AUC greater than 90% in several datasets querying approximately 10% (or less) of the data. In contrast, methods that achieved more than 90% AUC more frequently, such as RF initialized with 10 or 20% of the data, tended to query more labels. It is valuable to notice that the results for HebbAL were obtained by updating the MLP kernel after each newly labeled instance became available, which led to a slight improvement in the model's predictive performance. However, we found that it is equally feasible to use the model without updating the kernel, particularly when a larger number of labeled instances is available prior to the start of online training. In fact, our experiments indicated that HebbAL with a fixed kernel achieves significantly lower execution times than all other evaluated models on most datasets, with only a marginal reduction in AUC.

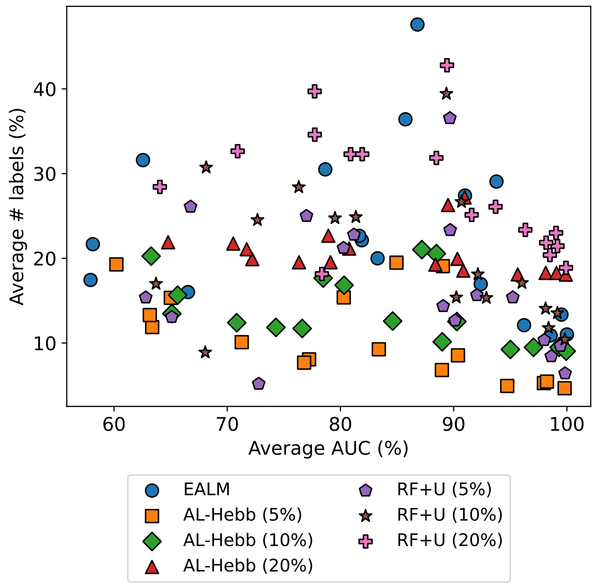

Fig. 3. Mean AUC and fraction of the data used by the models in each dataset.

A key limitation for the performance of HebbAL lies in its need for a large initial labeled set to train the MLP kernel. Since the benchmarks consisted mainly on small data problems, even 20% of the training data could be too little to train a neural network with reasonable flexibility. If the model cannot obtain a suitable representation for the data in the likelihood space, two serious problems arise: the input data of the Hebbian perceptron can be nonlinear and the crosstalk can be large. Also, applying the model in small data problems required updating the kernel with every queried sample, so that the learned projection could be improved over the continual training of the model. This is also non-ideal, since retraining the neural kernel for every queried label adds computational overheads to the algorithm and undermines the benefits of Hebbian learning. Therefore, future work could benefit from larger datasets with larger initial labeled sets, so that a suitable likelihood space could be learned and fixed before starting the online active training. In this case, updating the model with Hebb's rule presents substantial advantages: once a suitable likelihood space is learned and fixed, the kernel no longer needs to be retrained, and newly labeled samples can be incorporated through simple additive updates to the Hebbian classifier's weights. Another promising direction involves adapting the complexity of the neural kernel to the number of labeled samples available prior to online training. Techniques based on Structural Risk Minimization (SRM) [15] are particularly suitable here, as they incorporate model complexity into the training objective to prevent overfitting. In the context of MLPs, this can be achieved by selecting solutions that minimize training error while keeping the L2 norm of the synaptic weights small, effectively regularizing the network. For instance, Rocha et al. [16] proposed an algorithm that efficiently searches for low-error solutions under norm constraints. Also, the update of the neural kernel could be optimized to reduce the computational efforts in scenarios with limited initial supervision. A key challenge in this context is that standard online updates require running several backpropagation iterations over the entire labeled dataset whenever a new sample becomes available, since retraining the model only with a new example typically leads to forgetting previously acquired knowledge. In fact, this is a very general problem in machine learning known as catasthrophic forgetting [17]. To address this, several online learning methods for neural networks were proposed. One widely adopted strategy is Elastic Weight Consolidation (EWC) [18], which mitigates forgetting by penalizing changes in weights that would make the error grow in previously learned samples. By employing such approaches, the neural kernel could be trained more efficiently, since only the newest sample would be presented to it at each step. Finally, the kernel could also be updated only when some performance metric drops below a predefined threshold, avoiding unnecessary updates and improving overall efficiency.

VI. Conclusion

We proposed HebbAL, an active learning model based on a Hebbian classifier operating over kernel-based embeddings of data. In our experiments across 17 UCI benchmarks, HebbAL achieved competitive performance when compared to existing approaches to active learning, such as the widely used Random Forest with uncertainty querying. Although its average AUC was slightly lower, it consistently queried fewer labels during online training – a desirable feature in active learning scenarios. The main limitation of our method lies in the need for a sufficiently large pool of initial labeled data to learn a neural kernel that maximizes the distance between class clusters in likelihood space. When the kernel training is successful, the data becomes linearly separable and the crosstalk between samples of opposite classes is minimized, so a classifier can be trained using Hebb's Rule. This allows for continual learning within computational constraints, since updating a Hebbian model is straightforward and inexpensive. Therefore, future works include applying HebbAL to larger datasets, where more labeled samples are available prior to online training and an appropriate kernel can be learned. We believe this direction can leverage the benefits of Hebbian training and allow for efficient and scalable online active learning.

References

[1] J.-N. Vittaut, M.-R. Amini, and P. Gallinari, "Learning Classification with Both Labeled and Unlabeled Data," in *Machine Learning: ECML 2002*, T. Elomaa, H. Mannila, and H. Toivonen, Eds. Berlin, Heidelberg: Springer, 2002, pp. 468–479.

[2] R. M. Monarch, *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI.* Simon and Schuster, 2021.

[3] H. Yu, C. Sun, W. Yang, X. Yang, and X. Zuo, "AL-ELM: One uncertainty-based active learning algorithm using extreme learning machine," *Neurocomputing*, vol. 166, pp. 140–150, Oct. 2015. [Online]. Available: https://doi.org/10.1016/j.neucom.2015.04.019

[4] S. M. Shankaranarayana, "Model Uncertainty based Active Learning on Tabular Data using Boosted Trees," Oct. 2023.

[5] D. Cacciarelli and M. Kulahci, "Active learning for data streams: A survey," *Machine Learning*, vol. 113, no. 1, pp. 185–239, Jan. 2024.

[6] T. A. Ushikoshi, E. J. Freitas, M. Menezes, W. J. Junior, L. C. Torres, and A. P. Braga, "Hebbian learning with kernel-based embedding of input data," *Neural Processing Letters*, vol. 56, no. 6, p. 250, 2024.

[7] D. O. Hebb, *The organization of behavior: a neuropsychological theory.* J. Wiley; Chapman & Hall, 1949.

[8] M. V. Menezes, L. C. Torres, and A. P. Braga, "Width optimization of RBF kernels for binary classification of support vector machines: A density estimation-based approach," *Pattern Recognition Letters*, vol. 128, pp. 1–7, 2019. [Online]. Available: https://doi.org/10.1016/j.patrec.2019.08.001

[9] M. V. F. Menezes, "Learning Representations for Classification Problems in Reproducing Kernel Hilbert Spaces," Ph.D. dissertation, Universidade Federal de Minas Gerais, 2020.

[10] E. G. Horta, C. L. D. Castro, and A. P. Braga, "Stream-Based Extreme Learning Machine Approach for Big Data Problems," *Mathematical Problems in Engineering*, vol. 2015, pp. 1–17, 2015.

[11] J. A. Hertz, A. S. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, ser. Santa Fe Institute. Boulder: Chapman and Hall/CRC, 2018, no. v.1.

[12] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *2004 IEEE International Joint Conference on Neural Networks*, vol. 2, Jul. 2004, pp. 985–990 vol.2.

[13] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *IEEE transactions on electronic computers*, no. 3, pp. 326–334, 1965.

[14] M. Fernández-Delgado, J. Ribeiro, E. Cernadas, and S. B. Ameneiro, "Direct parallel perceptrons (dpps): fast analytical calculation of the parallel perceptrons weights with margin control for classification tasks," *IEEE transactions on neural networks*, vol. 22, no. 11, pp. 1837–1848, 2011.

[15] V. N. Vapnik, *The Nature of Statistical Learning Theory.* New York, NY: Springer New York, 2000. [Online]. Available: http://link.springer.com/10.1007/978-1-4757-3264-1

[16] H. P. Rocha, M. A. Costa, and A. P. Braga, "Neural networks multiobjective learning with spherical representation of weights," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4761–4775, Nov 2020.

[17] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, Apr. 1999, publisher: Elsevier. [Online]. Available: https://www.cell.com/trends/cognitive-sciences/abstract/S1364-6613(99)01294-2

[18] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwińska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.