Machine learning as ecology

Machine Learning as Ecology

Owen Howell,[1,*] Cui Wenping,[1,2] Robert Marsland III,[1] and Pankaj Mehta[1,†]

[1]*Department of Physics, Boston University, 590 Commonwealth Ave., Boston, MA 02215, USA*
[2]*Department of Physics, Boston College, 140 Commonwealth Avenue, Chestnut Hill, MA 02467*

(Dated: May 12, 2020)

Machine learning methods have had spectacular success on numerous problems. Here we show that a prominent class of learning algorithms - including Support Vector Machines (SVMs) – have a natural interpretation in terms of ecological dynamics. We use these ideas to design new online SVM algorithms that exploit ecological invasions, and benchmark performance using the MNIST dataset. Our work provides a new ecological lens through which we can view statistical learning and opens the possibility of designing ecosystems for machine learning.

INTRODUCTION

Machine learning (ML) is one of the most exciting and useful areas of modern computer science [1, 2]. One common machine learning task is classification: given labeled data from one or more categories, predict the category of a new, unlabeled data point. Another common task is to perform outlier detection (i.e. find data points that appear to be irregular). Both of these difficult problems can be solved efficiently using kernel-based methods such as Support Vector Machines (SVMs) [1, 3, 4].

The basic idea behind SVMs is to use a non-linear map to embed the input data in a high-dimensional feature space where it can be classified using a simple linear classifier (see Figure 1). To ensure good generalization and avoid overfitting, SVMs focus on the "hardest to classify" points that lie closest to the linear decision surface in the high-dimensional feature space. These points are called "support vectors" and play a prominent role in SVM algorithms.

The real power and utility of SVMs comes from the fact that these ideas can be implemented quickly and efficiently using kernel methods and quadratic optimization [1, 4]. The idea of a kernel function is to replace the *explicit* mapping to a high-dimensional feature space with an *implicit* kernel function that specifies the similarity (dot product) between data points in the high-dimensional feature space. Once the kernel function is specified, the support vectors and decision surface can be easily computed as an instance of a Quadratic Programming (QP) problem. There exist efficient exact and approximate optimization algorithms for QP that scale weakly polynomially in input size.

The original motivation for SVMs and other kernel methods were deep results in statistical learning theory concerning generalization errors [3–5]. Here, we show that these statistical problems can also be understood using ideas from niche theory in community ecology (see Table I) [6, 7]. Our construction exploits the recently discovered duality between ecological dynamics and constrained optimization problems, specifically quadratic programming [8–10]. In particular, we show

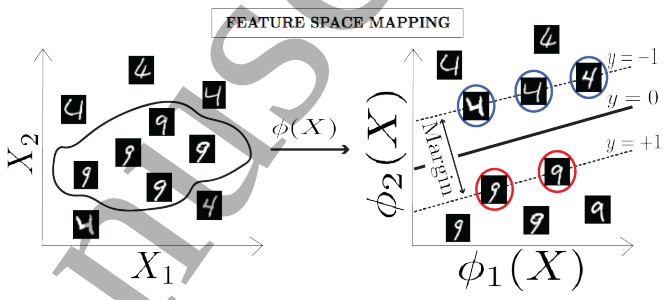

FIG. 1: Overview of Support Vector Machines (SVMs). Data points are mapped into a high-dimensional feature space via $\phi(X)$ where they can be separated using a linear decision surface. The SVM tries to maximize the distance (margin) from the decision boundary to the nearest data point. Points that lie on the maximum-margin planes (circled) are called support vectors and used to classify new, unlabeled data.

that data points can be viewed as "species" that compete for resources, with each feature identified with a distinct resource, and the kernel function specifying the niche overlap between species/datapoints [11, 12].

This mapping allows us to reinterpret SVMs as complex ecosystems that self-organize into ecologically stable steady states defined by their support vectors. This new ecological perspective naturally leads to a new *online* algorithms based on ecological invasion for SVMs as well as for outlier detection kernel methods such as Support Vector Data Description (SVDD)[13, 14]. We also show that our ecological SVDD method is equivalent to the online algorithm derived in [15].

SVMS AS QP

Consider a classification problem where each p-dimensional data point x_i $(i = 1, 2, 3 \dots N)$ comes with a binary label $t_i = \pm 1$. A SVM fits a linear classifier to the data of the form

$$y(x) = w^T \phi(x) + b \tag{1}$$

where $\phi : \mathbb{R}^p \to \mathbb{R}^q$, $q \gg p$ denotes a mapping to a high-dimensional feature space. The scalar offset b and the

q-dimensional weight vector w are tunable model parameters.

A new data point x_k is assigned to class $t_k = +1$ if $y(x_k) > 0$ and to class $t_k = -1$ if $y(x_k) < 0$. To begin, we restrict our discussion to linearly separable datasets, i.e., datasets for which exists a hyperplane in the feature space $\phi(x)$ that partitions the dataset into two regions with every point in class $+1$ in one region and every point in class -1 in the other (see Fig 1).

SVMs are trained by maximizing the margin, defined as the Euclidean distance from the hyperplane $y(x) = 0$ (the decision boundary) to the nearest data point. It is easy to show that distance from the point x_i to the line $y(x) = 0$ is given by the expression $t_i \frac{y(x_i)}{|w|}$. Maximizing the margin corresponds to choosing the parameters w and b so that

$$w, b = \arg\max_{w,b} \left\{ \frac{1}{|w|} \min_i [t_i (w^T \phi(x_i) + b)] \right\} \quad (2)$$

The above maximization problem can be recast by noting that Equation (2) has a gauge degree of freedom: the decision surface is invariant under the scaling transformation $w \to Dw$ and $b \to Db$ [1, 3, 4]. We can fix this gauge by choosing the margin to be exactly 1. In this gauge, Equation (2) is equivalent to the following convex quadratic programming problem

$$\begin{aligned} &\arg\min_{w,b} \frac{1}{2}|w|^2 \\ &\text{subject to } t_i(w^T\phi(x_i) + b) \geq 1 \text{ for all } i, \end{aligned} \quad (3)$$

where i labels the N data points in the training dataset.

As with all constrained optimization problems, we can also solve the equivalent dual optimization problem by introducing generalized Lagrange multipliers a_i (often called KKT multipliers in the optimization literature) corresponding to each of the inequality constraints in (3) [16]. Since there is one constraint per data point i, we can uniquely associate each a_i with a data point in the training set. For data points that saturate the inequality in (3), a_i is positive, and acts as an ordinary Lagrange multiplier to enforce the constraint. For the rest of the data points, no Lagrange multiplier is required, and $a_i = 0$. These observations give rise to the Karush-Kuhn-Tucker conditions, which are necessary and sufficient to determine the optimum [1, 3, 4]:

$$\begin{aligned} 0 &= \nabla_{w,b} L(w, b, a_i) \\ 1 &\leq t_i(w^T\phi(x_i) + b) \\ 0 &\leq a_i \\ 0 &= a_i[t_i(w^T\phi(x_i) + b) - 1] \end{aligned} \quad (4)$$

where the last three expressions hold for all i, with the SVM Lagrangian

$$L(w, b, a_i) = \frac{1}{2}|w|^2 - \sum_{i=1}^{N} a_i[t_i(w^T\phi(x_i) + b) - 1]. \quad (5)$$

Solving the first condition for w and b yields the equations $w = \sum_{i=1}^N a_i t_i \phi(x_i)$ and $\sum_{i=1}^N a_i t_i = 0$. Inserting these results into Equation (5) gives equations for optimal a_i:

$$\begin{aligned} &\text{argmax}_{a_i}\ L(a_i) = \sum_{i=1}^{N} a_i - \frac{1}{2}\sum_{i,j=1}^{N} a_i a_j t_i t_j K(x_i, x_j) \\ &\text{subject to } 0 \leq a_i \text{ for all } i \\ &\text{and } \sum_{i=1}^{N} a_i t_i = 0 \end{aligned} \quad (6)$$

$L(a_i)$ is called the dual SVM Lagrangian. In writing this equation, we have introduced the kernel function $K(x_i, x_j) \equiv \phi^T(x_i)\phi(x_j)$ which is just the dot product of the data points in the high-dimensional feature space ϕ. In this work we will specifically focus on the case where $K(x_i, x_j) \equiv \phi^T(x_i)\phi(x_j) \geq 0$. This is reasonable as many commonly used kernels are positive definite. For example, the Radial Basis Function kernel defined as $K(x_i, x_j) = \exp(\frac{-1}{2\sigma^2}||x_i - x_j||^2)$ is often used in the machine learning community [17].

In this dual formulation, the support vectors correspond precisely to those data points x_k for which the corresponding KKT multiplier is greater than zero $a_k > 0$. The SVM can be used to classify a new point x using $t = \text{sign}(y(x))$ with

$$\begin{aligned} y(x) &= \sum_{i \in S} t_i a_i K(x, x_i) + b \\ b &= \frac{1}{|S|}\sum_{i \in S}\left[t_i - \sum_{j \in S} a_j t_j K(x_i, x_j) \right] \end{aligned}$$

and S the set of support vectors.

THE ECOLOGY OF SVMS

Consider the maximization of the dual Lagrangian $L(a_i)$ given in Equation (6), subject to the constraints $\sum_{i=1}^N a_i t_i = 0$ and $a_i \geq 0$. Recently, it was shown there exists a duality between constrained optimization and ecological dynamics [8]. Using this duality, it is straightforward to show that the solution to this problem is encoded in the steady state of a generalized Lotka-Volterra equation of the form

$$\begin{aligned} \frac{da_i}{dt} &= a_i \left[1 + \lambda t_i - \sum_{j=1}^{N} t_i t_j K(x_i, x_j) a_j \right] \\ \frac{d\lambda}{dt} &= -\sum_{i=1}^{N} a_i t_i, \end{aligned} \quad (7)$$

This system of differential equations has a natural ecological interpretation as the dynamics of N species with

SVM	Ecology
Data point	Species
KKT Multiplier	Species Abundance
Feature Space	Trait Space
Kernel	Niche Overlap
Support Vectors	Species that survive in ecosystem

TABLE I: Conceptual mapping between SVMs and ecology

abundances a_i $(i = 1 \dots N)$ whose interactions are represented by the matrix α_{ij} with elements

$$\alpha_{ij} = t_i t_j K(x_i, x_j). \tag{8}$$

Since each a_i corresponds to a data point, we can think of this as an ecological network where data points i and j from the same class ($t_i = t_j$) compete with each other (i.e. $\alpha_{ij} > 0$) whereas species of from different classes ($t_i = -t_j$) are mutalistic (i.e. $\alpha_{ij} < 0$). The level of competition or mutualism depends on the overlap kernel $K(x_i, x_j)$ with similar data points having stronger interactions.

Ecologically, a combination of competitive and mutalistic interactions such as these naturally occur in plant-pollinator networks [18]. In such networks, different species of plants compete with each other for pollinators, pollinators compete with each other for plants, and plant-pollinators interactions are beneficial for both kinds of species. The λ term, which is the Lagrange multiplier for the constraint $\sum_{i=1}^{N} t_i a_i$, corresponds to an abiotic environmental factor that is produced or consumed by different species. In this plant-pollinator analogy, λ could represent an environmental CO_2 concentration. Specifically, plants consume CO_2 and benefit from high CO_2 concentration while pollinators produce CO_2 and are harmed by high CO_2 concentration.

Note that this interpretation differs from the consumer-resource interpretation given to a generic constrained optimization problem in [8]. The Lagrange multiplier λ plays the role of a "resource," but is not required to be positive, since it is enforcing an equality constraint rather than an inequality. The variables a_i of the optimization are now treated as the species rather than as resources.

These observations suggest a new ecological interpretation of SVMs (see Table I). Data points act like species that either compete or promote each others' survival. The abundance of each species is the value of KKT multiplier that enforces the corresponding constraint in (3). Since only the support vectors have non-zero KKT multipliers, the only data points that survive in the ecosystem are support vectors. As noted above, data points from the same category compete with each whereas data points from different categories are mutualistic. As is widely appreciated in the ecological literature, the ecological dynamics depends only on the overlap of resource utilization function encoded in the similarity kernel $K(x_i, x_j)$ between points [11]. The data points most likely to survive in the ecosystem are data points from one category that are similar to data points from the opposite category since they have large mutualistic interactions. For this reason, the data points that survive in the ecosystem are precisely those lie near the boundary between the two categories, that is, the support vectors.

ECOSVM: AN ONLINE SVM ALGORITHM

One interesting class of processes that has been extensively studied in the ecological literature is ecological invasion [9, 19–21]. In the context of SVMs, invasion by new species corresponds to addition of a new data point (x_0, t_0) to our existing dataset. If we denote the existing support vectors by the set S, the condition for a successful invasion is the intuitive statement that the initial growth rate must be positive when the new data point is introduced into the ecosystem:

$$0 < \frac{1}{a_0}\frac{da_0}{dt} = 1 + \lambda t_0 - \sum_{j \in S} t_0 t_j K(x_0, x_j) a_j. \tag{9}$$

The variable λ may be eliminated from this equation. Specifically, let $a_k > 0$ be a support vector. Then,

$$0 = 1 + \lambda t_k - \sum_{j \in S} t_k t_j K(x_k, x_j) a_j.$$

solving for λ gives

$$\lambda = -t_k + \sum_{j \in S} t_j K(x_k, x_j) a_j.$$

Inserting this equation for λ back into equation (9) gives

$$0 < \frac{1}{a_0}\frac{da_0}{dt} = 1 - t_k t_0 - \sum_{j \in S} t_0 t_j (K(x_k, x_j) - K(x_0, x_j)) a_j. \tag{10}$$

When this equation is satisfied the new data point can successfully invade the ecosystem and fixate (i.e. become a support vector). If the condition is not satisfied, the point goes "extinct" and the set of support vectors does not change. If a data point can invade successfully, the species abundances "a_i" are modified and can be found by solving for the steady state of (7) using either forward integration, quadratic programming [8] or any other online SVM approximation scheme [22–27]. This suggests a simple new approximate algorithm for online SVM learning we term the EcoSVM. In online learning, rather than seeing all the data at once, training data is presented in a sequential pattern. In the EcoSVM algorithm when a new training data point is presented, the invasion condition (10) is used to determine whether it can successfully invade the ecosystem. If it cannot, the training data

point is discarded. If it can, we recompute the steady-states using Equation (7). We will also generalize this algorithm to the case of non-separable data.

Algorithm 1 EcoSVM psuedocode

procedure INITIALIZE($D = \{(x_i, t_i)_{i=1}^{N_s}\}$)
  $a_i \leftarrow \text{argmax}_{a_i}\ L(a_i)$
    with a_i subject to $0 \leq a_i$ for all $i = 1, 2...N_s$
    and $\sum_{i=1}^{N_s} a_i t_i = 0$
  $S = \{(a_i, x_i, t_i) | a_i > 0\}$
  Return S
procedure ECOSVM($(x, t), S$)
  Invasion $= (1 - t_1 t - \sum_{j \in S} t t_j (K(x_1, x_j) - K(x, x_j)) a_j)$
  If Invasion < 0:
    Return S
  If Invasion > 0:
    $D = \{(x, t)\} \cup \{(x_i, t_i) | (a_i, x_i, t_i) \in S\}$
    $S =$ Intialize(D)
    Return S

Psuedocode for the EcoSVM algorithm: The subroutine Initialize takes a dataset $D = \{(x_i, t_i)_{i=1}^{N_s}\}$ (N_s is the size of the dataset.) as an input and returns $S = \{(a_i, x_i, t_i) | a_i > 0\}$, the set of KKT multipliers $a_i > 0$ greater then zero and the corresponding data points x_i and labels t_i. The routine EcoSVM takes a new data point (x, t) and the set of active support vectors and data points $S = (a_i, x_i, t_i)$ and computes the new set of support vectors using the criterion (10).

Because we use the ecologically inspired invasion condition, there is no need to recompute the support vectors at each learning step, unlike the online SVM algorithms that were previously suggested [22–27]. The EcoSVM algorithm also reduces the amount of training data that needs to be stored in memory. Specifically, instead of needing to store all data points, we keep only the support vectors. Since the number of support vectors is in general a small subset of all the training data, this greatly reduces the memory requirements. This increased efficiency comes at the expense of introducing small errors that come from the contingent nature of ecological invasions. Occasionally, a successful invasion by a new species (new data point) will allow a species that could not previously invade the ecosystem (designated not a support vector) to become viable (a support vector). This kind of historical contingency introduces errors in our online algorithm since we discard all data points that do not fixate in the ecosystem. In practice, we find that these errors are generally quite small for real-world datasets.

ADDING THE SLACK

In the previous sections we have focused on datasets that are linearly separable. For the majority of practical applications this is not the case. For overlapping class distributions, the primal SVM problem is modified so that points are allowed to be on the wrong side of the margin. Specifically, slack variables $\zeta_i \geq 0$ are introduced with $t_i y(x_i) \geq 1 - \zeta_i$. This should be compared with the linearly separable case where the constraint is instead $t_i y(x_i) \geq 1$. The new minimization is weighted to penalize points that lie on the wrong side of the margin

$$\begin{aligned} &\arg\min_{w,b,\zeta_i}\ \frac{1}{2}|w|^2 + C\sum_{i=1}^{N}\zeta_i \\ &\text{subject to, for all } i \\ &t_i(w^T\phi(x_i) + b) \geq 1 \\ &t_i y(x_i) \geq 1 - \zeta_i \\ &\zeta_i \geq 0 \end{aligned} \tag{11}$$

where the slack parameter C determines the extent to which points on the wrong side of the margin are tolerated. In practice, C is a hyper-parameter that is tuned to minimize generalization error. The KKT conditions for this new minimization problem are:

$$\begin{aligned} 0 &= \nabla_{w,b,\zeta_i} L(w, b, \zeta_i, a_i, \mu_i) \\ 1 - \zeta_i &\leq t_i y(x_i) \\ 0 &\leq a_i \\ 0 &= a_i[t_i y(x_i) - 1 + \zeta_i] \\ 0 &\leq \zeta_i \\ 0 &\leq \mu_i \\ 0 &= \mu_i \zeta_i \end{aligned} \tag{12}$$

where the μ_i are additional KKT multipliers enforcing the constraints $\zeta_i \geq 0$, and the primal Lagrangian is

$$\begin{aligned} &L(w, b, \zeta_i, a_i, \mu_i) = \\ &\frac{1}{2}|w|^2 + \sum_{i=1}^{N}[C\zeta_i - a_i(t_i y(x_i) - 1 + \zeta_i) - \mu_i\zeta_i] \end{aligned}$$

Minimizing the Lagrangian $\frac{\partial L}{\partial w_i} = 0$, $\frac{\partial L}{\partial b} = 0$ and $\frac{\partial L}{\partial \zeta_i} = 0$ gives equations

$$w = \sum_{i=1}^{N} t_i a_i \phi(x_i)\ ,\ \sum_{i=1}^{N} a_i t_i = 0\ ,\ a_i = C - \mu_i \tag{13}$$

Each $\mu_i \geq 0$ so the last equation is equivalent to $a_i \leq C$. Inserting these results into the primal Lagrangian transforms the problem into maximization of the dual SVM Lagrangian

$$\begin{aligned} &\text{argmax}_{a_i}\ L(a_i) = \sum_{i=1}^{N} a_i - \frac{1}{2}\sum_{i,j=1}^{N} a_i a_j t_i t_j K(x_i, x_j) \\ &\text{subject to } 0 \leq a_i \leq C \text{ for all } i \\ &\text{and } \sum_{i=1}^{N} a_i t_i = 0 \end{aligned} \tag{14}$$

We can enforce the second constraint by introducing a Lagrange multiplier λ, resulting in the following set of equations for the optimal a_i:

$$\begin{aligned} &\text{argmax}_{a_i,\lambda}\ L(a_i,\lambda) \\ &\text{subject to } 0 \le a_i \le C \text{ for all } i \end{aligned} \tag{15}$$

with Lagrangian

$$L(a_i,\lambda) = \sum_{i=1}^{N} a_i - \frac{1}{2}\sum_{i,j=1}^{N} a_i a_j t_i t_j K(x_i,x_j) + \lambda \sum_{i=1}^{N} t_i a_i \tag{16}$$

Using the duality described in [8], we can map the quadratic programming problem (15) to ecological dynamics

$$\begin{aligned} \frac{da_i}{dt} &= a_i(C-a_i)(1+\lambda t_i - \sum_{j=1}^{N} t_i t_j K(x_i,x_j) a_j) \\ \frac{d\lambda}{dt} &= -\sum_{i=1}^{N} a_i t_i. \end{aligned} \tag{17}$$

where the prefactor $a_i(C-a_i)$ enforces the constraints on a_i. Equation (17) has a similar interpretation to the Lotka-Volterra equations for the linearly separable case, with the additional $(C-a_i)$ factor can be interpreted as each species having a maximum carrying capacity C [28].

Now consider the addition of new point $P_0 = (x_0, t_0)$. This point changes the set of support vectors if the initial growth rate is positive.

$$0 < \frac{1}{a_0}\frac{da_0}{dt} \propto 1 + \lambda t_0 - \sum_{j=1}^{N} t_0 t_j K(x_0,x_j) a_j \tag{18}$$

Let x_k be any "active" support vector, that is, a point whose KKT multiplier a_k satisfies $C > a_k > 0$. Then, solving the steady state equation (17) for auxiliary variable λ gives

$$\lambda = -t_k + \sum_{i=1}^{N} t_i K(x_i,x_k) a_i \tag{19}$$

Inserting this into Equation (18) gives the invasion condition

$$0 < \frac{1}{a_0}\frac{da_0}{dt} = 1 - t_k t_0 + \sum_{i=1}^{N} t_i t_0 (K(x_i,x_k) - K(x_i,x_0)) a_i \tag{20}$$

This invasion condition can be used to construct an online learning algorithm. Specifically, when a new data point is presented, the condition (20) can be used to determine whether the new point changes the set of support vectors without having to recompute the minimum of (14). The nonzero KKT multipliers $a_i > 0$ and corresponding support vectors x_i are kept in memory. Note that (20) is identical to the invasion criterion (10) in the linearly separable case. The only difference between our algorithm for linearly separable and non-linearly separable datasets is due to the presence of the $a_i \le C$ constraint in the non-linear problem when the invasion condition is satisfied and the new steady state must be recomputed (Compare (6) and (14)).

A new point x is classified using $t = \text{sign}(y(x))$ with

$$\begin{aligned} y(x) &= \sum_i t_i a_i K(x,x_i) + b \\ b &= \frac{1}{|M|}\sum_{i\in M}\left[t_i - \sum_{j\in S} a_j t_j K(x_i,x_j)\right] \end{aligned}$$

where S is the full set of support vectors and M is the subset of active support vectors. Note that this formula requires at least one active support vector.

PERFORMANCE ON TOY MODELS

We test our proposed online learning algorithms on two toy datasets. We consider one dataset that is linearly separable in the feature space $\phi(x) = x$. Specifically, we choose all data points to be drawn from the $[0,1]^p$ p-dimensional hypercube. We then define the decision surface:

$$B_1 : (x_1 = \frac{1}{2}, x_2, ...x_p)$$

We consider a second dataset that is not linearly separable. Specifically, we define the second dataset to have decision boundary given by:

$$B_2 : (x_1 = \frac{1}{2} + \frac{1}{10}\sin(2\pi x_2)\sin(2\pi x_3)...\sin(2\pi x_p), x_2, ...x_p)$$

To test our proposed algorithm, we draw N total points from the p dimensional hypercube. First, we start our algorithm by computing the minimum of the SVM Lagrangian for the first N_s data points with $N_s \ll N$. We require that N_s is commensurate with p or else there are flat directions and our algorithm can become unstable. At each step, a new point is presented and the invasion condition (18) is used to determine whether the set of support vectors is changed. If (18) is satisfied, the steady state is recomputed using quadratic programming. This is continued for all N points. We find an excellent agreement between the predictions of our online algorithm and a batch SVM trained using all N points for both the linearly separable and non-linearly separable datasets.

We study how the test accuracy and number of support vectors depend on the training epoch T. For this purpose, let us define the accuracy

$$A(T) = 1 - \frac{1}{|N_{test}|}\sum_{x\in N_{test}} \frac{1}{2}|t_T(x) - t_{Exact}(x)|$$

where N_{test} is the set of testing data and $t_{Exact}(x)$ is the true label corresponding to point x. $t_T(x)$ denotes the prediction of the online SVM trained with T data points. In addition, let us define

$$N(T) = \begin{cases} \text{Number of } a_i(T) > 0 \\ \text{ for linearly seperable case} \\ \text{Number of } C > a_i(T) > 0 \\ \text{ for non-linearly seperable case} \end{cases}$$

where $a_i(T)$ are the support vector coefficients of the online SVM trained with T data points. In both the linear and non-linear case $N(T)$ counts the number of active support vectors.

Figure 2 and 3 show that in both the linear and non-linear case for large T the online algorithm converges to an accuracy $A(T)$ that is just below the accuracy of a batch SVM. Furthermore the number of active support vectors that the the online method finds after training is slightly below the number of support vectors that the batch SVM has.

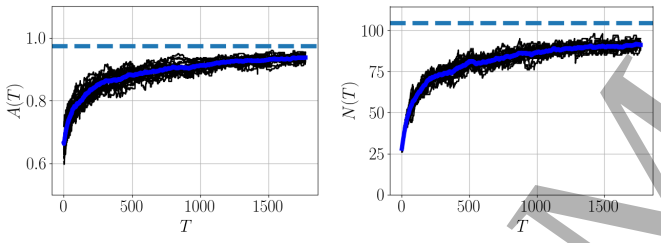

FIG. 2: Test accuracy and number of support vectors as a function of time for the linearly separable toy model, with true decision boundary B_1 defined in the SI text. Left panel shows accuracy of online SVM algorithm $A(T)$ as a function of the number of points T that the online SVM has seen. Black lines show individual realization, blue line shows mean accuracy. Dotted blue line shows batch SVM accuracy. Right panel shows the number of support vectors $N(T)$ as a function of the number of points T. Black lines show individual realization, blue line shows mean number of support vectors. Dotted blue line shows number of support vectors in SVM trained on entire dataset at once. The dimension of the data space is $p = 100$ and the online training is initialized with $N_s = 30$ data points.

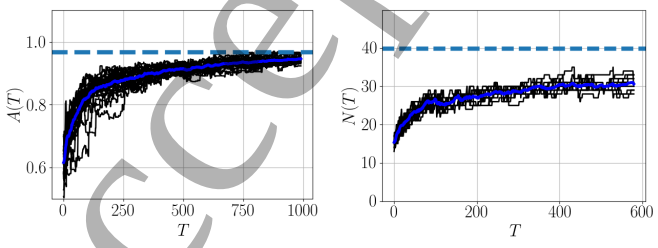

FIG. 3: Same as Figure 2, but for the non-linearly separable toy model, with true decision boundary B_2. The dimension is $p = 30$ and the initial number of points is $N_s = 30$.

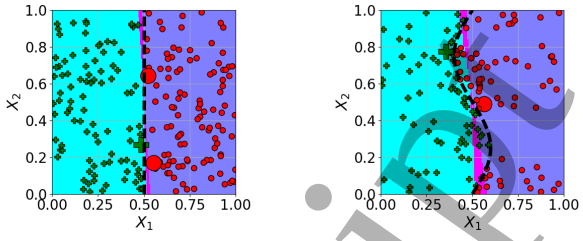

FIG. 4: Comparison of classical SVM and EcoSVM algorithms for a data set with $N = 200$ total training points points for (Left) a linearly separable dataset and (Right) a dataset where the data is not linearly separable (see section for more detail on algorithm in non-linear case). The true decision boundary is given by dashed black line. Cyan regions show data that the batch SVM and online SVM both identify as $t_i = 1$. Blue regions show data that batch SVM and online SVM both identify as $t_i = -1$. Purple regions show area in which SVM and online SVM disagree. $t_i = 1$ data points are shown as green plus symbols, $t_i = -1$ data points are shown as red circles. Active support vectors are shown with larger symbols.

Figure 4 shows the decision boundaries found by the EcoSVM were very similar to those found using an ordinary batch SVM algorithm in the $p = 2$ case.

We also compare the time complexity of an EcoSVM with a standard SVM. The time complexity of a quadratic programming problem of instance size N is $O(N^2)$. Thus, the time complexity of an online SVM is $O(\sum_{k=1}^{N} k^2) = O(N^3)$. A SVM in a p-dimensional space will generically have p active support vectors. Ergo, the time complexity of EcoSVM is roughly $O(\sum_{k=1}^{N} p^2) = O(Np^2)$. Thus, the EcoSVM has a factor of $O(\frac{N^2}{p^2})$ speedup over a traditional SVM. Figure 5 shows the CPU time as a function of the number of data points N for a standard online batch SVM and an EcoSVM. The left panel of figure 6 shows the test accuracy $A(N)$. Note that the EcoSVM is $O(N^2)$ faster and that the accuracies of the traditional SVM and EcoSVM are the same within about three percent. Furthermore, the left panel of figure 6 shows that EcoSVM and a traditional SVM find the same number of active support vectors.

PERFORMANCE ON MNIST

Next, we tested the performance of EcoSVM algorithm on MNIST [29, 30], a standard benchmark dataset in machine learning. The MNIST dataset consists of 6,000 training images and 1,000 test images of each of the handwritten digits '0'-'9'. To test the EcoSVM, we considered the binary classification task of distinguishing fours and nines. For this classification problem, we used a standard Gaussian (RBF) kernel given by $K_\sigma(x, y) = \exp\left[-\frac{1}{2\sigma^2}(x - y)^2\right]$, where the kernel width σ was determined via cross validation on the batch SVM.

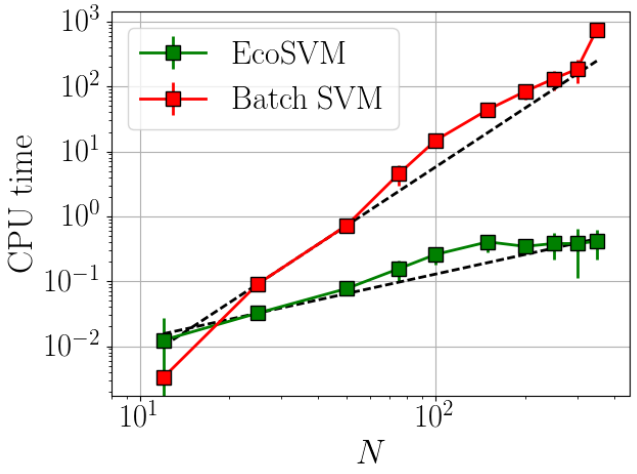

FIG. 5: System CPU time as a function of the number of points N for a batch SVM (red) and an EcoSVM (green). The black dotted lines are the theoretical scaling. The dimension is $p = 10$. Error bars are shown. Each SVM is started with 10 data points. Averaged over 100 realizations.

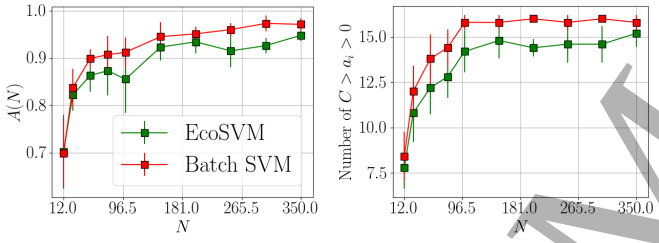

FIG. 6: Left panel shows the test accuracy $A(N)$ as a function of the number of points N for a traditional SVM (red) and an EcoSVM (green). Right panel shows the number of active support vectors a_i such that $C > a_i > 0$ as a function of the number of points N for a traditional SVM (red) and an EcoSVM (green). The dimension is $p = 10$. Error bars are shown. Each SVM is started with 10 data points. Error bars are shown. Averaged over 100 realizations.

The performance of our EcoSVM algorithm was comparable to a traditional SVM trained on the full dataset (98.1% accuracy compared to 98.5% accuracy for traditional SVMs, as shown in Figure 7). The EcoSVM algorithm also ends up finding a similar number of support vectors as a traditional SVM: ~ 750. We note that since the MNIST dataset is not completely linearly separable in the RBF feature space, in these numerical simulations we used the generalization of the EcoSVM algorithm for non-linearly separable datasets.

ECOLOGY TO SVDD

Support Vector Data Description (SVDD) is an unsupervised learning algorithm closely related to SVMs that

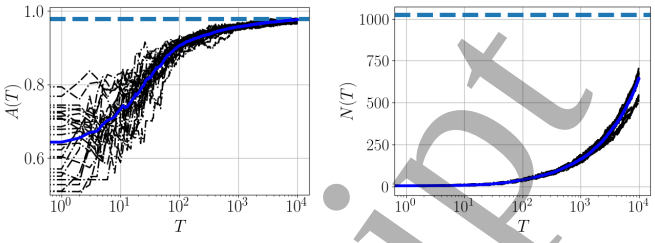

FIG. 7: An ecologically inspired online SVM algorithm EcoSVM applied to digit classification of nines and fours from the MNIST dataset [29, 30]. The right panel shows the accuracy of EcoSVM for 25 different realizations. The average accuracy over all realizations is shown using the solid blue line and the accuracy of an SVM trained on the entire dataset is shown using the blue dotted line. The right panel shows the number of active support vectors in each realization (black lines), the mean number of support vectors across all realizations (blue solid line) and the number of support vectors for SVM trained on full dataset (dotted blue line).

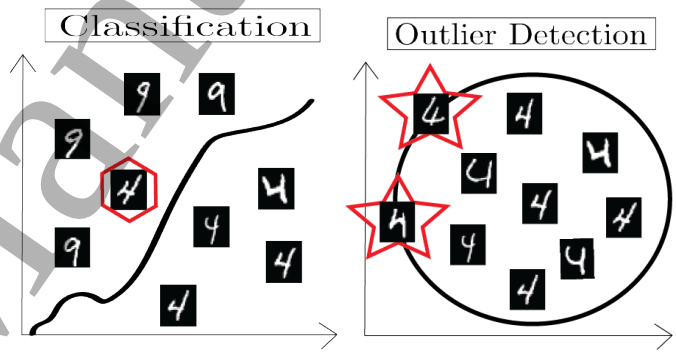

FIG. 8: Schematic showing instances of a supervised classification problem (left) and an outlier detection problem (right). In the supervised classification problem the goal is to partition the space into two distinct volumes, one for each label. The thick black line denotes the decision boundary. An incorrectly classified four (inside the red hexagon) is shown on the wrong side of the decision boundary. In the outlier detection problem, a sphere is created in the feature space to enclose the minimum volume while still containing all data points (in this case fours). Points on the boundary of this sphere are called outliers, they are starred in the schematic. The thick black line denotes the sphere boundary.

performs outlier detection (see Figure 8 and [13, 14] for a good overview). The SVDD problem consists in finding a sphere of minimum radius in some feature space that contains all the data points. Points that lie on the surface of the sphere are called outliers and are analogous to active support vectors in SVMs (see Figure 8). We show below reinterpreting SVDDs in terms of ecological dynamics naturally leads to an online SVDD learning algorithm identical to the one found in Jiang et al. in 2017 [15].

We can mathematically formulate the problem as follows. Given a set of unlabeled data points $\mathcal{D} = (x_i)_{i=1}^N$,

find the sphere of minimum radius R that contains all the data points:

$$\begin{aligned} &\text{minimize } R^2 \\ &\text{subject to } |\phi(x_i) - \mu|^2 \leq R^2 \text{ for all } i \end{aligned} \tag{21}$$

where the function $\phi(x)$ defines the feature space and μ is the center of the sphere. This problem is simplified by the introduction of KKT multipliers a_i for each inequality constraint. The KKT conditions for the minimization problem are:

$$\begin{aligned} 0 &= \nabla_{R,\mu} L(R, \mu, a_i) \\ 0 &\leq a_i \\ |\phi(x_i) - \mu|^2 - R^2 &\leq 0 \\ a_i[|\phi(x_i) - \mu|^2 - R^2] &= 0 \end{aligned} \tag{22}$$

with

$$L(R, \mu, a_i) = R^2 + \sum_{i=1}^{N} a_i(|\phi(x_i) - \mu|^2 - R^2)$$

Minimizing $L(R, \mu, a_i)$ with respect to μ and R gives equations:

$$\sum_{i=1}^{N} a_i = 1 \quad \text{and} \quad \mu = \sum_{i=1}^{N} a_i \phi(x_i) \tag{23}$$

Substituting Equation (23) into Equation (22) gives maximization problem for the optimal a_i:

$$\begin{aligned} &\text{argmax}_{a_i} \; L(a_i) = \sum_{i=1}^{N} a_i K(x_i, x_i) - \sum_{i,j=1}^{N} a_i a_j K(x_i, x_j) \\ &\text{subject to } 0 \leq a_i \text{ for all } i \\ &\text{and } \sum_{i=1}^{N} a_i = 1 \end{aligned} \tag{24}$$

$L(a_i)$ is called the dual SVDD Lagrangian and $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$. For simplicity we set $K(x_i, x_i) = 1$ for the diagonal elements in the rest of the derivation, although the results we present are easily generalized for arbitrary kernel. Python code, which can be found at https://github.com/owenhowell20/EcoSVM, works for any choice of $K(x, y)$.

After the SVDD is trained, the sphere radius in feature space R can be determined via

$$\begin{aligned} R^2 &= \text{max}_i \; |\phi(x_i) - \mu|^2 \\ &= \text{max}_i \; (\phi(x_i) - \mu)^T (\phi(x_i) - \mu) \\ &= \text{max}_i \; (\phi(x_i)^T \phi(x_i) - 2\mu^T \phi(x_i) + \mu^T \mu) \end{aligned} \tag{25}$$

The explict dependence on $\phi(x)$ in (25) can be removed using $\mu = \sum_{i=1}^{N} a_i \phi(x_i)$:

$$\begin{aligned} &\phi(x_i)^T \phi(x_i) - 2\mu^T \phi(x_i) + \mu^T \mu = \\ &1 - 2\sum_{j=1}^{N} K(x_i, x_j) a_j + \sum_{j,k=1}^{N} K(x_j, x_k) a_j a_k \end{aligned}$$

where we have used $\phi(x_i)^T \phi(x_j) = K(x_i, x_j)$ and $K(x_i, x_i) = 1$. Thus, (25) can be written in terms of kernel function and support vectors as

$$\begin{aligned} R^2 &= \text{max}_i \big[\\ &1 - 2\sum_{j=1}^{N} K(x_i, x_j) a_j + \sum_{j,k=1}^{N} K(x_j, x_k) a_j a_k \big] \end{aligned} \tag{26}$$

We apply our method to the SVDD Lagrangian (24). As $K(x_i, x_i) = 1$ and $\sum_{i=1}^{N} a_i = 1$ the first term in the sum can be ignored. A Lagrange multiplier λ is introduced to enforce the latter constraint. The quantity to be maximized is then

$$\begin{aligned} &\text{argmax}_{a_i, \lambda} L(a_i, \lambda) = -\sum_{i,j=1}^{N} a_i a_j K(x_i, x_j) + \lambda(\sum_{i=1}^{N} a_i - 1) \\ &\text{subject to } 0 \leq a_i \text{ for all } i \end{aligned} \tag{27}$$

Using the quadratic programming-ecology duality, we can embed the solution to (24) as the steady state of the dynamical equations

$$\begin{aligned} \frac{da_i}{dt} &= a_i(\lambda - \sum_{j=1}^{N} K(x_i, x_j) a_j) \\ \frac{d\lambda}{dt} &= 1 - \sum_{i=1}^{N} a_n \end{aligned} \tag{28}$$

Now, suppose we are at the steady state of Equation (28) and consider the addition of a new point x_0. The invasion condition is that the initial growth rate is positive

$$0 < \frac{1}{a_0} \frac{da_0}{dt} = \lambda - \sum_{j=1}^{N} K(x_0, x_j) a_j \tag{29}$$

Let x_k be any point which has non-zero support vector, $a_k > 0$. Then, solving the steady state equation (28) for auxiliary variable λ gives

$$\lambda = \sum_{i=1}^{N} K(x_i, x_k) a_i \tag{30}$$

Inserting this into Equation (29) gives the invasion condition

$$0 < \frac{1}{a_0} \frac{da_0}{dt} = \sum_{i=1}^{N} a_i [K(x_k, x_i) - K(x_0, x_i)] \tag{31}$$

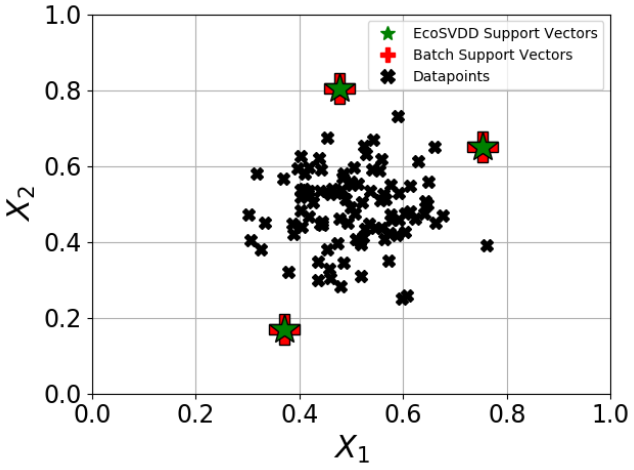

FIG. 9: Comparison of batch SVDD and online SVDD algorithms for a data set with $N = 100$ total points (shown in black) drawn from a Gaussian distribution. Batch SVDD active support vectors are shown with red "+" symbols. EcoSVDD active support vectors are shown with green stars. The kernel function is Gaussian $K(x, y) = \exp(\frac{-1}{2}(x-y)^T(x-y))$. The EcoSVDD algorithm was started with 10 points.

which is identical to Equation (2.9) derived in [15] (Note that they use notation z for our variable x_0). Equation (31) can be used to formulate an online learning algorithm in the same manner as the EcoSVM algorithm presented in the main text. The paper by Jiang et al. derives Equation (31) and calls the algorithm Fast Incremental Support Vector Data Description (FISVDD). They also numerically show that this algorithm performs well on real-world datasets. The fact that this algorithm can be constructed simply and elegantly using a duality between quadratic programming and ecology suggests that there is a deeper connection between machine learning and ecological dynamics than previously realized and, more significantly, that ecologically inspired machine learning models are not just of theoretical interest but can be used for real world data analysis.

We illustrate the ability of FISVDD/EcoSVDD numerically. We draw data points from a p-dimensional multinomial Gaussian distribution with identity covariance matrix and mean uniformly sampled from the p-dimensional hypercube $x \in [0, 1]^p$. Figure (9) shows the two-dimensional case.

We define $R(T)$ to be the SVDD radius (26) of an FISVDD/EcoSVDD trained on T points. The left panel of Figure 9 shows $R(T)$ as a function of T. $R(T)$ converges to the batch SVDD radius (shown with a dashed blue line).

Similarly, we define a fidelity metric between the FISVDD/EcoSVDD kernel sphere center trained on T points $\mu(T)$ (23) and the batch SVDD sphere center $\tilde{\mu}$ as

$$S(T) = \frac{\mu(T)^T \tilde{\mu}}{\sqrt{\mu(T)^T \mu(T)}\sqrt{\tilde{\mu}^T \tilde{\mu}}} \quad (32)$$

$S(T) \leq 1$ with equality if and only if $\mu(T) = \tilde{\mu}$. The right panel of Figure 10 shows $S(T)$ as a function of T. $S(T)$ converges to 1 illustrating the fact that the FISVDD/EcoSVDD and batch SVDD produce the same kernel sphere center and radius.

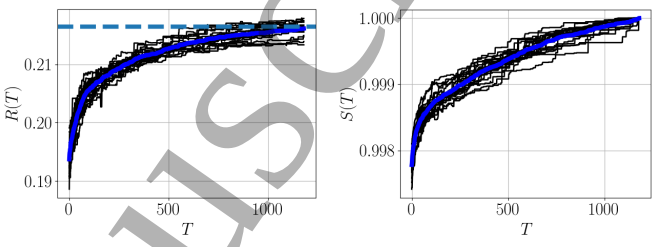

FIG. 10: Online SVDD radius and kernel sphere center similarity score as function of T. Left panel shows online SVDD radius $R(T)$ as a function of the number of points T that the online SVM has seen. Black lines show individual realization, blue line shows mean radius. Dotted blue line shows batch SVDD radius. Right panel shows the normalized dot product between $\mu(T)$ and $\tilde{\mu}$. Black lines show individual realizations, blue line shows average over realizations. The dimension of the data space is $p = 15$ and the online training is initialized with $N_s = 30$ data points.

CONCLUSION

In this work, we have shown how we can think about kernel methods using ideas from ecology. This ecological mapping allowed us to formulate a new ecologically inspired online SVM algorithm, the EcoSVM. We have shown that the performance of EcoSVM is comparable to traditional SVMs that work with all data simultaneously. Our algorithm differs from all previous online SVMs which we know of which must recompute the support vectors at each learning step [22–27]. As a result, our algorithm performs significantly faster for large datasets. For a dataset with N data points, our algorithm is typically faster by a factor of N^2. We also show that the equivalent online algorithm for outlier detection using SVDD is equivalent to the Fast Incremental SVDD algorithm found by [15].

Our results suggest that ecological dynamics may provide a rich new setting for thinking about biologically-inspired machine learning. The fact that algorithms with significant speed-up and much smaller memory requirements can be constructed simply and elegantly using a duality between quadratic programming and ecology suggests that there is a deeper connection between machine

learning and ecological dynamics than previously realized. It also suggests that ecologically inspired machine learning models are not just of theoretical interest but may be useful for real world data analysis.

Another promising research direction is the idea is to explore how we can implement complicated statistical learning tasks using real ecosystems. Our work suggests the tantalizing possibility that it maybe possible to engineer synthetic ecosystems that implement sophisticated statistical learning algorithms [31]. In particular, it will be interesting to ask if and how we can harness synthetic biology and CRISPR-based biological techniques to implement kernel methods.

Code implementing the EcoSVM and EcoSVDD algorithms can be found at https://github.com/owenhowell20/EcoSVM.

ACKNOWLEDGMENTS

OH acknowledges support from BU UROP student funding. The work was supported by NIH NIGMS grant 1R35GM119461, Simons Investigator in the Mathematical Modeling of Living Systems (MMLS) to PM.

* Electronic address: olh20@bu.edu
† Electronic address: pankajm@bu.edu

[1] C. M. Bishop, *Pattern recognition and machine learning* (springer, 2006).
[2] P. Mehta, M. Bukov, C.-H. Wang, A. G. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, Physics Reports (2019).
[3] C. Cortes and V. Vapnik, Machine learning **20**, 273 (1995).
[4] B. Schölkopf, A. J. Smola, F. Bach, *et al.*, *Learning with kernels: support vector machines, regularization, optimization, and beyond* (2002).
[5] V. Vapnik, *The nature of statistical learning theory* (Springer science & business media, 2013).
[6] R. MacArthur, Theoretical Population Biology **1**, 1 (1970).
[7] P. Chesson, Theoretical Population Biology **37**, 1 (1990).
[8] P. Mehta, W. Cui, C.-H. Wang, and R. Marsland, Phys. Rev. E **99**, 052111 (2019).
[9] R. MacArthur and R. Levins, The American Naturalist **101**, 3949 (1967).
[10] M. Advani, G. Bunin, and P. Mehta, arXiv preprint arXiv:1707.03957 (2017).
[11] R. MacArthur and R. Levins, The American Naturalist **101**, 377 (1967).
[12] R. K. Colwell and D. J. Futuyma, Ecology **52**, 567 (1971).
[13] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99 (MIT Press, Cambridge, MA, USA, 1999) pp. 582–588.
[14] D. M. J. Tax and R. P. W. Duin, Mach. Learn. **54**, 45 (2004).
[15] H. Jiang, H. Wang, W. Hu, D. Kakde, and A. Chaudhuri, in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33 (2019) pp. 3991–3998.
[16] S. Boyd and L. Vandernberghe, *Convex Optimization* (Cambridge University Press, 2004).
[17] C. M. Bishop, *Pattern Recognition and Machine Learning* (Springer, 2011).
[18] F. Valdovinos, P. Moisset de Espans, J. Flores, and R. Ramos-Jiliberto, Oikos **122** (2012), 10.1111/j.1600-0706.2012.20830.x.
[19] D. Tilman, Resource competition and community structure **17** (1982).
[20] K. Shea and P. Chesson, Trends in Ecology and Evolution **17**, 170 (2002).
[21] T. J. Case, Proceedings of the National Academy of Sciences **87**, 9610 (1990), https://www.pnas.org/content/87/24/9610.full.pdf .
[22] T. Poggio and G. Cauwenbergus, Advances in Neural Information Processing Systems (2000).
[23] N. Karampatziakis and J. Langford, arXiv preprint arXiv:arXiv:1011.1576 (2011).
[24] P. Laskov, C. Gehl, S. Krüger, and K.-R. Müller, J. Mach. Learn. Res. **7**, 1909 (2006).
[25] D. M. Tax and P. Laskov, in *Neural Networks for Signal Processing, 2003. NNSP'03. 2003 IEEE 13th Workshop on* (IEEE, 2003) pp. 499–508.
[26] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, in *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 80, edited by J. Dy and A. Krause (PMLR, Stockholmsmssan, Stockholm Sweden, 2018) pp. 4393–4402.
[27] F. Sohrab, J. Raitoharju, M. Gabbouj, and A. Iosifidis, arXiv:1802.03989 (2018).
[28] F. Valdovinos, E. Berlow, P. M. d. Espanes, R. Ramos-Jiliberto, D. P. Vazquez, and N. D. Martinez, Nature Physics **9**, 424 (2018).
[29] Y. LECUN, http://yann.lecun.com/exdb/mnist/ .
[30] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, in *Proceedings of the IEEE* (1998) pp. 2278–2324.
[31] A. R. Zomorrodi and D. Segre, Journal of molecular biology **428**, 837 (2016).

Supplemental Material

HYBRID ALGORITHM

The algorithm proposed in the main text can be modified to a hybrid algorithm that interpolates between a pure batch SVM and our EcoSVM. Specifically, instead of using the invasion condition (9) to determine if the set of active support vectors changes, quadratic programing can be used at every step to recompute the set of active support vectors. Inactive support vectors are then discarded.

The time speedup of our EcoSVM algorithm vs the hybrid algorithm depends on the underling distribution of

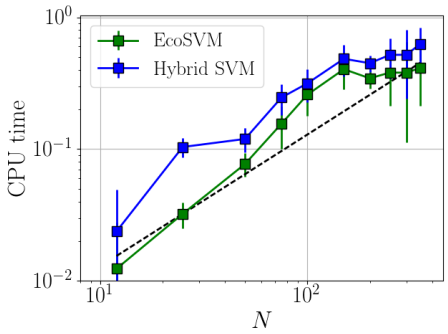

FIG. 11: System CPU time as a function of the number of points N for an EcoSVM (green) and a "Hybrid SVM" (blue). The black dotted lines are the theoretical scaling. The dimension is $p = 10$. Error bars are shown. Each SVM is started with 10 data points. Averaged over 100 realizations. The underling data distribution is the linear dataset described in section . Note that the difference in performance between the EcoSVM and HybridSVM depends on the underling data distribution.

the dataset. Consider the following thought experiment: if each added data point does not change the set of active support vectors, then the invasion condition in (9) is never satisfied and our algorithm can be made to run infinity faster then the hybrid algorithm (and the batch SVM). We expect that for generic distributions, including real datasets, the majority of new data points will not change the set of active support vectors, thus giving our algorithm a significant increase in speed over the hybrid algorithm.