Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

A MODELING-BASED CLASSIFICATION ALGORITHM VALIDATED WITH SIMULATED DATA

Karen Hovsepian
Peter Anselmo
Subhasish Mazumdar

Computer Science Department
New Mexico Tech
801 Leroy Place
Socorro, NM 87801, U.S.A.

ABSTRACT

We present a Generalized Lotka-Volterra (GLV) based approach for modeling and simulation of supervised inductive learning, and construction of an efficient classification algorithm. The GLV equations, typically used to explain the biological world, are employed to simulate the process of inductive learning. In addition, the modeling approach provides a key advantage over the more conventional constraint and optimization-based classification algorithms, as influences of outliers and local patterns, which can lead to problematic overfitting, are auto-moderated by the model itself. We present the bare-bones algorithm and motivate the model through axiomatic postulates. The algorithm is validated using benchmark simulated datasets, showing results competitive with other cutting-edge algorithms.

1 INTRODUCTION

Within the expansive field of Artificial Intelligence, supervised inductive machine learning is a crucial area concerned with algorithms for classification of observations based on existing observation/class example pairs. The term induction is due to this learn-from-examples framework. The observations can be either simulated or real, and a good supervised inductive machine learning algorithm is one that generalizes from the provided examples with most confidence and least error. Such accurate classification can provides usable information that otherwise would be difficult and costly to obtain.

The field of supervised inductive machine learning has seen powerful and robust algorithms, with good generalization characteristics, as in the example of the well-known Support Vector Machines (Cortes and Vapnik 1995), the Genetic Algorithm (Mitchell 1996), Artificial Neural Networks (Abdi, Valentin, and Edelman 1999). Most of these algorithms are constraint-based, seeking to minimize the error on the dataset of provided examples, also known as the training dataset. The motivation for training (empirical) error minimization is due to the well-known Empirical Risk Minimization (Vapnik and Chervonenkis 1989) learning principle. Some robust algorithms from the list of abovementioned algorithms are also optimized, seeking to minimize the error in a way that maximizes the confidence of the classification decision.

We present a classification algorithm based on a model that simulates the induction process taking place in the space of training observations. The states of this model are what we call *classification confidences*, measuring the classification influences of the example observations. Our model describes the induction of these confidences and is based on the Generalized Lotka-Volterra (GLV) equations (Lotka 1925; Volterra 1926).

The original and the main context for GLV equations is the modeling of ecological systems, where the states of the model are population densities of the biological species in question. The flexibility of the GLV equations is that they can describe several types of interaction dynamics that take place within the ecological system, such as predation, competition (for resources), and mutualism/cooperation. The discrete-time GLV equation is defined as:

$$\alpha_t^{(i)} = \alpha_{t-1}^{(i)} + b_i \alpha_{t-1}^{(i)} \left(1 - c_i \alpha_{t-1}^{(i)} + \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{i,j} \alpha_{t-1}^{(j)} \right) \qquad (1)$$

where $\alpha_t^{(i)}$ is the population density of the i^{th} species during period t. The model contains fixed constants b_i and c_i, known as the "per capita growth rate" and the coefficient of within-species interaction, respectively. These specify the rate of free growth from period-to-period and the rate at which the density reaches its intrinsic limit, known as the carrying capacity of the ecosystem for that species, due to competition for resources by members of the species. Naturally, the full ecological model contains several GLV equations, each describing a single biological species.

The term $\sum_{\substack{j=1 \\ j\neq i}}^{N} a_{i,j}\alpha_{t-1}^{(j)}$ is the component of the model responsible for introducing interactions. The constants $a_{i,j}$ specify the sign and strength of interactions between every pair of species. When $a_{i,j}$ is negative, the species j has a negative effect on the growth of species i. If $a_{j,i}$ is also negative, we have an example of competition. Conversely, if both $a_{i,j}$ and $a_{j,i}$ are positive, the pair of species are in a mutualistic/symbiotic relationship. Another common interaction type is predation, when species i, the predator, has a negative effect on species j, the prey, while j has a positive effect on i.

The model we develop on the basis of the GLV equations is also centered on interactions, but in our context interactions are between training examples of the classification problem. Whereas the equilibrium in an ecological model signifies a steady-state where all surviving species coexist with each other, our model's equilibrium is a state where all viable data patterns have been consulted and taken into consideration, and a consensus on the classification confidences of the training observations has been reached.

The outline of the paper is as follows. In the first section we introduce the postulates that motivate the use of our adapted model to supervised inductive learning. Then we present the full induction model, following which we formulate the algorithm itself, including the various essential coefficients. Following that we present the decision function. Lastly, we present the results of validation experiments, and conclude with a summary of our approach and future work.

2 INDUCTION MODEL

The classification induction problem is defined as follows: given a set of training input vectors $X = \{x_i \in \mathbb{R}^n | i = 1,\dots,N\}$ and a set of training output scalars, i.e. class labels, $Y = \{y_i \in \{1,\dots,k\} | i = 1,\dots,N\}$; find the *decision function* $f(\alpha) : x^* \to \hat{y}$, where x^* is an unclassified input vector, and $\hat{y}$ is our estimation of x^*'s "true" output. In the framework of machine learning, the training algorithm is said to "choose" this decision function from the set of decision functions $F = \{f(x,\alpha) | \alpha \in \Lambda\}$ by computing/inducing the vector α, which uniquely parameterizes the decision function f.

2.1 Induction Postulates

At the core of our proposed model is the measure *classification confidence*, which is shortened in the paper to simply *confidence*. For training vectors, this quantity measures the strength of a vector's membership in its corresponding class — the greater it is, the more confident we are that the vector belongs to the class. In the context of inductive machine learning, the trained confidences are the induction parameters in vector α, mentioned earlier. Table 1 lists all the symbols associated with the confidence measure.

Confidences are the states of our model. Prior to the convergence of the model we consider the confidences to be partially trained, with the fully trained confidences found at the convergence of the model. In the context of machine learning methodology, confidences are the trainable parameters of our decision function —vector α. Thus, solving the convergence problem of our inductive model corresponds to training/choosing the decision function parameters.

The first group of postulates connects the modeling framework to the machine learning framework by presenting the model convergence problem and outlining how the trained confidences are utilized in the decision function.

Postulate 2.1 *Define a sequence $\left\{\alpha_t^{(i)}\right\}$ of partially learned confidences for i^{th} training vector, where t is the learning step index. The final learned confidence is found at the convergence limit of this sequence:*

$$\alpha_i = \lim_{t\to\infty} \alpha_t^{(i)} \tag{2}$$

Postulate 2.1 states that α_i, i.e. the trained confidence for x_i, is found at the convergence limit of the sequence $\left\{\alpha_t^{(i)}\right\}$. Consequently, induction is the process that leads to joint convergence of all confidences.

The model that describes the induction of the partially trained confidences is defined in postulate 2.2 as a system of first-order recurrence equations, one for each confidence.

Postulate 2.2 *Sequence $\left\{\alpha_t^{(i)}\right\}$ is formulated as a recurrence equation:*

$$\alpha_t^{(i)} = \alpha_{t-1}^{(i)} + g_i\left(A_{t-1}, X, Y\right), i = 1,\dots,N, \tag{3}$$

where $g_i\left(A_{t-1}, X, Y\right) \to \mathbb{R}$ is the recurrence formula for the i^{th} training vector.

We adopt the GLV equations, defined earlier, to formulate the recurrence equations $g_i(\cdots)$. The conceptual and philosophical motivations for using the GLV equations in this context are described in the next section.

Postulate 2.3 *Define a function $h_c(A,X,Y,x^*) : \mathbb{R}^N \times \mathbb{R}^{n^N} \times \mathbb{Z}^{+N} \times \mathbb{R}^n \to \mathbb{R}$ for each class label c, such that*

$$\widehat{\alpha_c} = h_c(A,X,Y,x^*). \tag{4}$$

Then, the predicted class of x^* is chosen by comparing the predicted classification confidences for all class labels:

$$\hat{y} = \underset{c_i}{\text{argmax}}(\widehat{\alpha_{c_i}} | i = 1,\dots,k). \tag{5}$$

Table 1: All symbols related to the classification confidence concept.

$\alpha_i \geq 0$	:	**trained** *classification confidence* of training vector x_i.
$A = \{\alpha_i \| i = 1, \dots, N\}$	:	vector of trained classification confidences of all training vectors.
$\alpha_t^{(i)}$	:	**partially trained** classification confidence of i^{th} training vector at induction step t.
$A_\tau = \left\{\alpha_\tau^{(i)} \| i = 1, \dots, N\right\}$	:	vector of partially trained classification confidences at step τ.
$\widehat{\alpha_c} \in \mathbb{R}$	:	**predicted** classification confidence of x^*, assuming it belongs to class c.

The class with the highest predicted confidence — the class that is predicted to be the best fit for x^* — is chosen as $\widehat{y}$.

2.2 Induction Model

Below we present the full formulation of the model used for inductive machine learning. As mentioned in the Introduction, the model uses GLV equations, due to applicability of the core concepts expressed by the GLV equations to the paradigm of inductive learning.

$$\alpha_t^{(i)} = \alpha_{t-1}^{(i)} + \alpha_{t-1}^{(i)} \left(1 - \alpha_{t-1}^{(i)} + \sum_{\substack{j=1 \\ j \neq i}}^{N} \delta_{i,j} s_{i,j} \alpha_{t-1}^{(j)} \right) \tag{6}$$

$i = 1, \dots, N,$

The model is based on two dynamic principles. The first is *logistic growth*, characterized by near-exponential increase of the confidences when they are small, and decay when they reach some critical size. In the context of ecological modeling, logistic growth allows finite convergence of the population densities to the carrying capacity of the ecosystem, caused by intra-species competition and scarcity of resources.

In our context, logistic growth is explained with the following postulate:

Postulate 2.4 (Maximal overfitting)

$$\forall i,t : \exists m > 0 : \frac{g_i(A_t, \cdots) - g_i(A_{t-1}, \cdots)}{\alpha_t^{(i)} - \alpha_{t-1}^{(i)}} \propto (m - \alpha_{t-1}^{(i)})$$

The maximal overfitting postulate 2.4 states that the slope of the recurrence function is proportional to the inflection point m minus the previous confidence $\alpha_{t-1}^{(i)}$. Thus, if confidence at the previous induction step was much smaller than m, the direction and rate of confidence change is positive and proportionally high. In contrast, if the previous confidence is above m, the next induced confidence will be lower.

Essentially this postulate describes how each training vector can attain its full potential confidence, if it were to completely decide for itself. Consisting of this principle alone, the model creates conditions for maximal overfitting, because each confidence is induced in isolation relying on the training vector alone, as if it were its own class. The part of the equation in (6) that implements the logistic growth principle is the term $\alpha_{t-1}^{(i)} \left(1 - \alpha_{t-1}^{(i)}\right)$, with m in Postulate 2.4 equal to 1.

The second dynamic principle of the model is interaction between the training examples. As with logistic growth [maximal overfitting principle], interaction is a key principle in the context of ecological modeling, affecting the growth/change of every species' population density during each discrete time period. The interaction effect can either be positive, as between cooperating/symbiotic organisms, or negative, as in the case of competition or harvesting/predation.

In our induction model, this principle corresponds to regularization of learning, by relating induction of each vector's confidence to the induction of all other vectors' confidences, thus moderating the maximal overfitting principle. The following two postulates motivate positive and negative interactions between the confidences of input vectors.

Postulate 2.5 (Negative Interaction)

$$\forall i,j,t : y_i \neq y_j \Rightarrow \frac{g_i(A_{t+1}, \cdots) - g_i(A_t, \cdots)}{\alpha_{t+1}^{(j)} - \alpha_t^{(j)}} < 0$$

The motivation for mutually negative interaction between vectors of opposing classes is that being more certain that patterns in one part of the space belong to a certain class implies that we are less confident in the classification of vectors from another class, situated near or around that part of the space.

Postulate 2.6 (Positive Interaction)

$$\forall i,j,t : y_i = y_j \Rightarrow \frac{g_i(A_{t+1}, \cdots) - g_i(A_t, \cdots)}{\alpha_{t+1}^{(j)} - \alpha_t^{(j)}} > 0$$

By contrast, postulate 2.6 states that vectors from the same class have mutually positive effects on the growths of each other's confidences, justified by the shared classification and also as a way to counteract the effect of negative interaction with vectors of other classes.

In the model formulation, the term $\sum_{\substack{j=1\\ j\neq i}}^{N} \delta_{i,j} s_{i,j} \alpha_{t-1}^{(j)}$ implements the interaction principle, with $\delta_{i,j}$ differentiating between positive and negative interactions. The model also uses the interaction strength coefficient $s_{i,j}$, defined for each pair of vectors, which quantifies the strength of the interaction. Greater $s_{i,j}$ allows the confidence of vector j to have greater effect on the rate and direction of change of confidence of vector i.

2.3 Induction Model Convergence

Before we discuss the convergence and equilibrium conditions of our model, it is convenient to switch from the discrete-time recurrence model to continuous-time differential model. The only difference introduced with this formulation is that now induction takes place every instant of time, whereby the confidences also change values continuously. To model this continuous, instantaneous time, induction, a system of differential equations takes the place of the system of recurrence equations. This means that equations (6) are rewritten as:

$$\frac{d\alpha_i}{dt} = \alpha_i \left(1 - \alpha_i + \sum_{\substack{j=1\\ j\neq i}}^{N} \delta_{i,j} s_{i,j} \alpha_j \right), \tag{7}$$

where α_i is defined as a function of time. This is a system of linear first-order differential equations. The convergence, or equilibrium, is found by setting the right-hand side to 0, and finding the roots (Meerschaert 1999). This results in the following linear system of equations:

$$A^I(1 - MA) = 0, \tag{8}$$

given a $N \times 1$ vector $A = \{\alpha_i | i = 1, \ldots, N\}$, an $N \times N$ diagonal matrix $A^I = \left\{ A^I_{i,i} = \alpha_i | i = 1, \ldots, N \right\}$, and an $N \times N$ matrix M, defined earlier.

The system (8) may have solutions where, where some, or indeed all, α_i's are 0, due to the term A^I. However, since the induction outcome with any training confidence equal to 0 is not preferred, we can focus our attention only on the parenthesized term, solving the following equation:

$$1 - MA = 0, \tag{9}$$

Solving (9) attains what is known in population ecology as the "community equilibrium", whereby all species coexist in an equilibrium. This is the system we first introduced in the beginning of the paper. To be sure that a "community equilibrium" exists, we must ascertain that $\det(M) \neq 0$.

The trained confidences are found by solving the linear non-homogeneous system (9). It's important that it is a stable equilibrium, in order to ascertain that the partially trained confidences will indeed converge to the fully trained confidences. A key theorem on systems of differential equations states that an equilibrium is stable if and only if all eigenvalues of the Jacobian of the left hand side of (8), evaluated at the equilibrium values, have negative real parts (Hirsch and Smale 1974). As a necessary condition for stability of the equilibrium the determinant cannot be allowed to be negative.

2.4 The Convergence Solution System

Based on the above discussion, we formally present the system of linear equations that represents the solution to our proposed model's convergence problem.

$$MA = -1, \tag{9}$$

where M is a full-rank $N \times N$ matrix, and A is a $N \times 1$ vector, corresponding to the aforementioned parameter vector α. We solve the system above in order to find the tuning parameters of our decision function; after that we can proceed to classify new observations. Without jumping too much ahead, we should note that the above system is the solution to the convergence problem of our stated induction model. Let us define M, which we call our *interaction matrix*:

$$M = \left\{ M_{i,j} | i, j = 1, \ldots, N \right\}, \tag{10}$$

with $M_{i,j} = \begin{cases} -1 & \text{if } i = j \\ \delta_{i,j} s_{i,j} & \text{otherwise.} \end{cases}$

The sign of every off-diagonal entry in M, given by the coefficient $\delta_{i,j}$, depends on whether the row-indexed input vector is from the same class as the column-indexed vector — $\delta_{i,j}$ is 1 when $y_i = y_j$ and -1 when $y_i \neq y_j$.

The absolute value of every off-diagonal entry is what we call the "interaction coefficient" $s_{i,j} \in (0, 1]$, unique not only for each i and j, but also for the order in which they are paired: $s_{i,j} \neq s_{j,i}$. We will present their full computation formulae in the next section.

2.5 Interaction Coefficients

The interaction coefficient $s_{i,j}$ is defined as the product of three multipliers. These are: "similarity", "relevance weight", and "class bias penalty".

$$s_{i,j} = [\text{similarity}]_{i,j} \times [\text{relevance weight}]_i \times [\text{class bias penalty}]_{y_j} \quad (11)$$

Kernels are central to M's construction. The similarity multiplier, as its name suggests, measures the similarity between i^{th} and j^{th} vectors. By using kernels to compute the similarity multiplier, we allow arbitrary levels of non-linearity, and hence capacity, for out classifier. In our work we have explored two kernels: the Radial Basis Function,

$$K(u,v) = e^{-\|u-v\|\sigma}, \quad (12)$$

and a kernel based on the hyperbolic tangent function,

$$K(u,v) = 1 - \tanh(\|u-v\|\sigma). \quad (13)$$

Both kernel functions morph all Euclidean distances into one of two polarities: 0 or 1, and σ modulates the degree of morphing. Prior to starting the training phase, the user chooses the kernel and the σ, which are used to compute the interaction coefficients. For example, for high σ's, most Euclidean distances are morphed into near-zero values, and only very short Euclidean distances are morphed into values close to 1.

$$[\text{similarity}]_{i,j} = K(x_i, x_j) \quad (14)$$

Besides the similarity multiplier, the interaction coefficient is based on the relevance weight multiplier, which also uses kernels:

$$[\text{relevance weight}]_i = \frac{\xi_i}{\sum_{\substack{j=1\\ y_j=y_i}}^{N} \xi_j} \frac{\zeta_i}{\sum_{\substack{j=1\\ y_j=y_i}}^{N} \zeta_j}, \quad (15)$$

where the quantities ξ_i and ζ_i, which can be loosely named "similarity to opposite classes" and "similarity to own class", are defined as:

$$\xi_i = \sqrt{\sum_{\substack{j=1\\ y_j \neq y_i}}^{N} (K(x_i, x_j))^2} \quad (16)$$

$$\zeta_i = \sqrt{\sum_{\substack{j=1\\ y_j = y_i}}^{N} (K(x_i, x_j))^2}. \quad (17)$$

The relevance weight multiplier is essentially the product of two ratios, quantifying the relative distance of the i^{th} vector to opposite and own classes, respectively. The weight takes on values between 0 and 1, with the more "relevant" or "important" vectors having relevance weights closer to 1.

The final multiplier used to define the interaction coefficients is the class bias penalty, computed on the basis of the distance to own class, defined above:

$$[\text{class bias penalty}]_{y_j} = 1 - \frac{\theta_{y_j}}{\sum_{c=1}^{k} \theta_c} \quad (18)$$

$$\theta_c = \frac{\sum_{\substack{i=1\\ y_i=c}}^{N} \zeta_i}{N_c}. \quad (19)$$

where N_c is the number of input vectors in class c. Essentially, θ_c is a measure of spread for class c.

The above definitions of the components of matrix M illustrate that it is not positive-definite. While kernel functions are central to its construction, matrix M cannot be categorized as a kernel Gram matrix, used in methods like Support Vector Machines and regularized least-squares. A kernel Gram matrix consists only of kernels. Matrix M on the other hand consists of coefficients, in which kernels serve a specific role — quantifying the similarity between vectors.

Once all interaction coefficients are computed and the matrix is constructed, the algorithm simply solves the system (9). The solution is performed by first computing the LU decomposition and solving two easier subproblems. This is a standard method used by most commercial solvers. With the solution to (9) found, we have the parameter vector of our decision function and we can proceed to classification of new unclassified vectors.

2.6 Heuristic Motivation of Interaction Coefficients

Here we present an explanation of the modeling logic and motivation for the use of kernels and the multipliers of the interaction coefficients (11). The similarity multiplier (14) is the basis of the interaction coefficient. The basic heuristic at work here is that more similar vectors should experience more interaction and have stronger effect on induction of each other's confidences. For this simple reason, kernels are ideal, since they attain values close to 1 for highly similar vectors and values closer to 0 for dissimilar vectors. For vectors of the same class, the rationale for stronger positive interaction resulting from more similarity is that increased sharing of patterns between vectors of the same class further cements our confidence that these shared patterns belong to their common class. For vectors of different classes, on the other hand, more similarity suggests that stronger competition should take place, since the patterns in such a small space can't belong to both classes at the same time.

Next we try to explain the basis for multiplying the similarity by the relevance weight. Broadly speaking, the goal is to adjust the raw potential interaction strength of

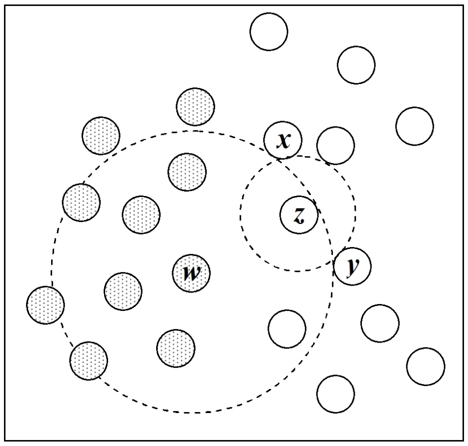


Figure 1: An example illustrating how vectors have unique priorities and relevances within their classes.

each vector x_i, which is given by the similarity multiplier, by a scalar that represents how important it is to interact with x_i. This is done because not all vectors in the dataset are as relevant as others. Some vectors play pivotal roles in assuring high confidences for other vectors in their classes, while others are so far from the 'action' that they are virtually irrelevant.

From the earlier section, the relevance weight is equal to a product of two ratios:

$$[\text{relevance weight}]_i = \frac{\xi_i}{\sum_{\substack{j=1 \\ y_j = y_i}}^{N} \xi_j} \frac{\zeta_i}{\sum_{\substack{j=1 \\ y_j = y_i}}^{N} \zeta_j} \tag{15}$$

The first ratio ranks, as a scalar between 0 and 1, x_i's relative similarity to other classes. The nearer a vector is to other classes, the more contentious it is, and since it represents the more contentious patterns, it gains priority with respect to all positive and negative interactions.

As an example of this prioritization, we refer to Figure 1. By virtue of its proximity to the opposite class, x plays an important role in assisting the confidences of vectors in its class, including those of the less contentious y and z. Hence, we need to concentrate z's positive effect on x at the cost of its effect on y, even though x and y are equidistant from it.

The second ratio in (15) ranks, as a scalar between 0 and 1, x_i's relative similarity to its own class. The purpose of this ratio is to more accurately compute the relevance weights for vectors that are situated far from their own classes. The reasoning is that a vector, even one situated near opposite classes, that does not represent its class very well cannot be as relevant as one that does. More specifically, this quantity would make sure that extreme outliers, i.e., vectors situated near other classes, but far from their own classes, do not end up with very large relevance weights.

In general terms, the relevance weight is defined on the basis of the joint assessment of how well the vector defines the contentious patterns and how well it defines the bulk of its own class, or the easier patterns. The intuitive result is that a highly relevant vector x_i is one that captures the difficult patterns, which however have a decent chance of being determined to belong to x_i's class.

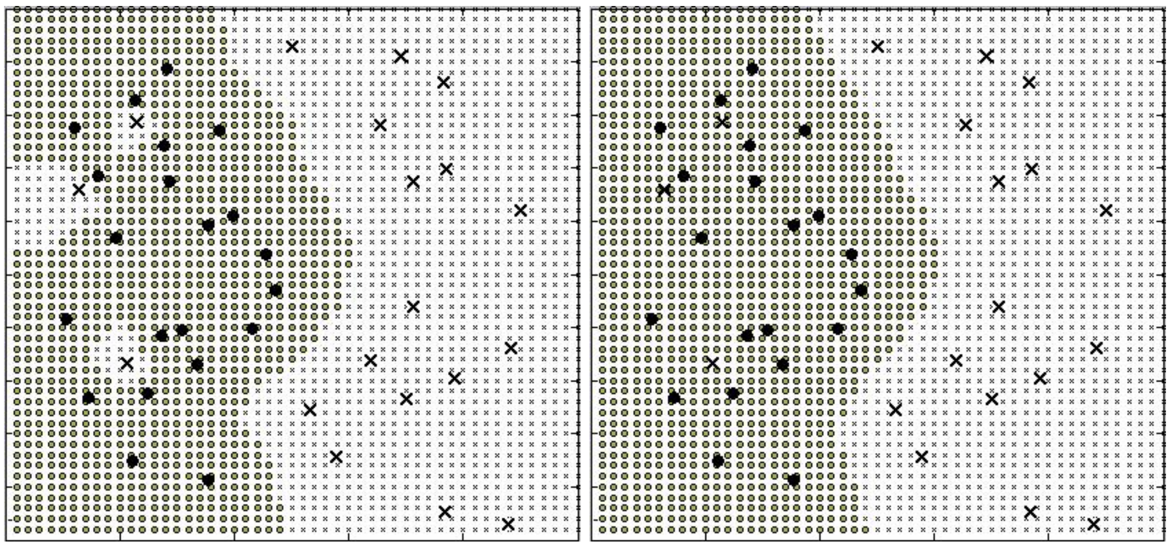
The third and final multiplier of the interaction coefficient is the class bias penalty multiplier. Class bias is determined by how large and how 'tight' the class is relative to others. The size of the class is simply the number of vectors in the class and the tightness is determined by the sum of similarities of all vectors in the class to all other vectors in the class. Loosely speaking vectors of larger and/or tighter classes enjoy more and/or stronger positive interactions. An artifact of this relative class superiority is that these vectors will attain higher confidences than vectors of other classes. Consequently, this class will enjoy "unfair competition" and will outclass all other vectors simply because the training dataset contains more or more tightly grouped examples of this class. It brings to mind the well-known maximal margin separation based classifiers, such as SVMs (Cortes and Vapnik 1995). The underlying principle is that to be as general as possible, the classifier should be as impartial in its treatment of the classes as possible. This is achieved with the optimization framework.

3 DEPLOYING THE DECISION FUNCTION

Once the system (9) has been solved and the learned confidences have been found, we use them to predict the class of a new vector x^*. As postulate 2.3 shows, in order to predict its class, we must first find the predicted confidences of x^*, for each class in the data.

The complete approach would be to add x^* and its assumed class to the training data, and relearn the confidences of all vectors (training plus the unlabeled), from amongst which we only use the confidence of the unlabeled vector. A more efficient approach however is to keep all the learned training confidences fixed, which is justifiable if we note that they are in a stable equilibrium — the state that is most robust under perturbations. With this in mind, we present the recurrence equation for $\widehat{\alpha}_{t\,|c}$, which can be seen as the sequence of semi-estimated confidences for x^*, under assumption that it belongs to class c.

$$\widehat{\alpha_{t+1}}_{|c} = \widehat{\alpha}_{t\,|c} + \widehat{\alpha}_{t\,|c_i} \left(1 - \widehat{\alpha}_{t\,|c} + \sum_{j=1}^{N} \delta^*_{j|c} s^*_{j|c} \alpha_j \right) , \tag{20}$$



(a) Example with three outliers; $\sigma = 0.5$ (b) Example with two outliers; $\sigma = 0.03$.

Figure 2: Example illustrating the implicit control of overfitting with σ.

where $\delta^*_{j|c}$, and $s^*_{j|c}$ are the identical to $\delta_{i,j}$ and $s_{i,j}$ in (6), except that vector x_i is replaced with x^*, and the class y_i is replaced with the assumed class c.

In a sense, the above equation does not have any modeled interaction, since α_i's are **constant** values at this point. All it exhibits is logistic growth, where the constant term $\sum_{j=1}^{N} \delta^*_{j|c} s^*_{j|c} \alpha_j$ either increases or decreases the upper-bound on $\widehat{\alpha}_{t|c}$. This means that the only factor that affects the difference between all the predicted confidences is this summation term. Consequently, we can define the predicted confidence of x^* for each assumed class as this summation:

$$\widehat{\alpha_c} = \sum_{j=1}^{N} \delta^*_{j|c} s^*_{j|c} \alpha_j. \tag{21}$$

The toy example in Figure 2 (next page) illustrates the role that σ plays in not only defining the overall nonlinearity of classification boundary, but also in the implicit control of outliers and overfitting. In Figure 3, a large σ ensures that all three outliers of the **cross-marks** class have high enough confidence that unlabeled data around them are classified as **cross-marks**. However, by reducing the value of σ, we allow interactions to take a bigger part in defining the induction outcome, which, due to their location, forces the three outliers to become ineffectual in the context of classifying unlabeled data. A large σ sets maximal overfitting (cf. postulate 2.4) as the main effective inductive principle of the model by reducing the interaction effect, which benefits outliers due to their difficult location. This can cause poor generalization by forcing classification of the space around outliers that does not fit the general patterns of the data.

Conversely, a small σ increases the visibility of vectors to each other, which increases the association and inter-dependence between the confidences. As a result, the overall distribution of the vectors can affect the confidence of such outliers, with the more drastically unusual vectors being dominated and attaining ineffective confidences at convergence. According to statistical learning theory, more inter-dependence and association between the vectors is preferred to more individualistic learning, because it makes induction more robust and less sensitive to noise and outliers.

In summary, the σ parameter combines capacity control and overfitting control by controlling how effective the model is in weeding out outliers and noise that degrade the conditioning of the system matrix.

4 PRELIMINARY EXPERIMENTAL VALIDATION

As mentioned earlier, the solution to the system (9) comes down to computing the LU decomposition, after which two easier systems are solved quickly. Our code relies on the block algorithm routines provided in the powerful Linear Algebra PACKage (LAPACK) library (Anderson et al. 1999).

To provide some preliminary experimental validation, we compare our algorithm in the context of a well-quoted classification algorithm evaluation paper by Meyer et al. (Meyer, Leisch, and Hornik 2003), which compares the performance of 18 popular modern algorithms, including

Table 2: Simulated benchmark datasets used in this study.

	# observations	**# features**
circle	1200	2
spirals	1200	2
twonorm	1200	20
threeorm	1200	20
ringnorm	1200	20

Support Vector Machines (Cortes and Vapnik 1995), Neural Nets (Abdi, Valentin, and Edelman 1999), Double Bagging (Hothorn and Lausen 2003), on 21 benchmark datasets, most taken from the UCI machine learning database (Blake and Merz 1998). In this paper we report results for the five simulated datasets found in the paper. Results of experiments on other benchmark sets and ongoing development of this approach can be gleaned from the following site: `http://www.nmt.edu/~karen/imbl`.

The five datasets are: **Circle** (a circle inside a square), **Spirals** (two noisy intertwined spirals), **twonorm** (two multinomial distributions), **threenorm** (two multinomial distributions for one class and a third multinomial distribution for the third class), and **ringnorm** (a small multinomial distribution inside a bigger one). The datasets are summarized in Table 2.

The experimental design was as follows. Each training dataset was randomly split into 100 unique training/testing set pairs. Each training set contained 200 examples, with the testing sets containing the remaining 1000 examples. The splits were made while maintaining the same ratio of examples in each of two classes. In order to find the optimal parameter, the algorithm was trained on the 2/3rds of the training set, while the accuracy was tested on the remaining 1/3rd. The optimal σ was chosen from the range $\{-10^2, -9.9^2, -9.8^2, \ldots, 5^2]$. Using the optimal σ we retrained the algorithm with the full training set and measured its accuracy on the testing set. This was repeated for all 100 pairs, and the average accuracy was recorded.

The results are presented in Table 3. Our algorithm is mentioned in the table under the alias **IMBL**, which stands for Interaction Modeling Based Learning. In addition to our quoted results we list the rates of top 11 algorithms; we refer the reader to (Meyer, Leisch, and Hornik 2003) for mean error rates of all 18 algorithms. As the table shows, our algorithm outperforms all other 18 algorithms on 2 datasets (**Spirals** and **twonorm**), comes in at second place on 2 other datasets (**threenorm** and **ringnorm**) and fourth on the last dataset (**Circle**).

Our algorithm requires just one user-tuned parameter, σ. Moreover, the execution time is not sensitive to the choice of σ. These two facts, coupled with the fact that our algorithm comes down to computation of several weights and solution to a linear system of equations, allowed us to quickly find the optimal σ.

5 CONCLUSIONS AND FUTURE DIRECTIONS

The main benefit of using a simulated learning model is that the fully trained decision function is obtained from the stable steady-state of the model, since at this point the parameters of the decision function attain their equilibrium values. Here is where the choice of recurrence equation models is important, due to the relatively simple computation of the steady-state. The second feature of our work is the modification of the model with coefficients based on spatial relationships between the observations. Because learning is not based on artificially specified criteria and constraints, but rather on an intuitive juxtaposition of interactions and vectors' effect on themselves, the issue of overfitting is handled "implicitly". Finally, the formulation is such that our algorithm is not limited to two classes, but is rather inherently capable of truly multi-class classification.

Another benefit of algorithm's simplified formulation of training is the zero dependence of the training time-complexity on parameter σ. In many constraint-based algorithms, different parameter values may affect the time-complexity and practical speed of the training phase. This subtle point means, among other things, that real-time estimation of the duration of batch jobs (due to varying parameters) is impossible. Because training in our algorithm comes down to a one-time computation of interaction coefficients and solving a system of equations, there are no worst or best cases.

There are several directions of improvement and extension that promise a more accurate and efficient algorithm. Currently, the main focus of our work is updating and extending the formulae for computing the spatially-dependent interaction coefficients, with emphasis on primarily accuracy, while keeping algorithmic complexity increase to a minimum. An exploration of alternative interaction strength formulae and computational implementations is ongoing.

REFERENCES

Abdi, H., D. Valentin, and B. E. Edelman. 1999. *Neural networks*. Thousand Oaks: Sage.

Anderson, E., Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. 1999. *Lapack users' guide*. 3 ed. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Table 3: Mean error rates of our algorithm, labeled "IMBL", and top 11 algorithms on the five simulated benchmark datasets. Least error rate are marked with an asterisk, while the second least rates are marked with a double asterisk.

	svm	**lda**	**nn**	**f.bruto**	**nnet**	**glm**	**rFrost**	**bagg**	**dbagg**	**lvq**	**IMBL**
circle	2.66*	49.48	5.88	4.05**	4.17	49.49	6.73	7.47	6.59	44.3	5.52
spirals	0.81	49.99	0.17**	7.90	3.37	49.99	2.43	2.71	2.21	46.28	0.08*
twonorm	2.82**	3.16	7.27	4.13	5.32	5.64	4.09	7.96	2.83	3.07	2.45*
threenorm	15.76	18.2	25.29	21.69	22.01	18.58	18.55	21.22	17.22	14.17*	15**
ringnorm	3.58*	38.75	40.87	9.06	30.47	39.07	5.92	11.93	11.57	38.07	4.69**

Blake, C., and C. Merz. 1998. Uci repository of machine learning databases.

Cortes, C., and V. N. Vapnik. 1995. Support-vector networks. *Machine Learning* 20:273–297.

Hirsch, M., and S. Smale. 1974. *Differential equations, dynamical systems, and linear algebra*. New-York: Academic Press.

Hothorn, T., and B. Lausen. 2003. Double-bagging: combining classifiers by bootstrap aggregation. *Pattern Recognition* 36 (6): 1303–1309.

Lotka, A. J. 1925. *Elements of physical biology*. Baltimore: Williams and Wilkins.

Meerschaert, M. M. 1999. *Mathematical modeling*. 2 ed. Academic Press.

Meyer, D., F. Leisch, and K. Hornik. March 2003. The support vector machine under test. *Neurocomputing* 55:169–186.

Mitchell, M. 1996. *An introduction to genetic algorithms*. Cambridge, MA: MIT Press.

Vapnik, V. N., and A. Chervonenkis. 1989. The necessary and sufficient conditions for consistency of the method of empirical risk minimization. In *Yearbook of the Academy of Sciences of the USSR on Recognition, Classification, and Forecasting* (2 ed.)., 217–249. Nauka.

Volterra, V. 1926. Variations and fluctuations of the number of individuals in animal species living together. In *Animal Ecology*, ed. R. N. Chapman, 409–448. New-York: McGraw-Hill.

AUTHOR BIOGRAPHIES

KAREN HOVSEPIAN is a PhD candidate in Computer Science working with Drs. Anselmo and Mazumdar. The focus of his research dissertation is machine learning algorithms. He has also worked on applications of machine learning in the area of finance and bioinformatics. His email address for these proceedings is <karen@nmt.edu>.

PETER ANSELMO is Associate Professor and Chairman of the Management Department at New Mexico Tech. He is also a Research Scientist at the New Mexico Tech-based Institute for Complex Additive Systems Analysis. His current research interests are in agent-based modeling and simulation, with applications to complex financial and organizational systems, and in intelligent-systems approaches to data modeling and analysis. His email address for these proceedings is <anselmo@nmt.edu>.

SUBHASISH MAZUMDAR is an Associate Professor of Computer Science and Adjunct Associate Professor of Management at New Mexico Tech. His current research interests include problems of integrity of distributed and mobile databases, the integration of pre-existing heterogeneous databases and documents, as well as a conceptual modeling approach to information systems and software development. He has received support for his research from Sandia National Laboratory and the National Science Foundation. He is a member of the ACM, IEEE Computer Society, and Sigma Xi. His email address for these proceedings is <mazumdar@cs.nmt.edu>.