Mutual Online Concept Learning for Multiple Agents

Jun Wang Les Gasser
Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign
Champaign, IL 61820, USA
{junwang4, gasser}@uiuc.edu

ABSTRACT

To create multi-agent systems that are both adaptive and open, agents must collectively learn to generate and adapt their own concepts, ontologies, interpretations, and even languages actively in an online fashion. A central issue is the potential lack of any pre-existing concept to be learned; instead, agents may need to *collectively design* a concept that is evolving as they exchange information. This paper presents a framework for *mutual online concept learning* (MOCL) in a shared world. MOCL extends classical online concept learning from single-agent to multi-agent settings. Based on the Perceptron algorithm, we present a specific MOCL algorithm, called the *mutual perceptron convergence algorithm*, which can converge within a finite number of mistakes under some conditions. Analysis of the convergence conditions shows that the possibility of convergence depends on the quality of the instances they produce. Finally, we point out applications of MOCL and the convergence algorithm to the formation of adaptive ontological and linguistic knowledge such as dynamically generated shared vocabulary and grammar structures.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems, Languages and structures, Coherence and coordination; I.2.6 [**Artificial Intelligence**]: Learning—*Concept learning*

Keywords

Online concept learning, Mutual learning, Perceptron algorithm, Language evolution, Ontology evolution

1. INTRODUCTION

The development of foundational principles for multi-agent systems has progressed significantly over the past twenty to thirty years. But one area that has fallen far short of the general advance is the topic of multi-agent concept formation, e.g., as a basis for communication[8, 16, 22]. The standard approach to communication in the multi-agent systems community is to establish a) shared protocol standards that define and fix the syntax (and some of the semantics) of communications, and b) shared representational ontologies that define and fix the communicative lexicon (and its semantics). KQML[11] and KIF[7] are such examples. Unfortunately, the effectiveness of both of these foundations is questionable:

1. Under this model, the elements of language (e.g., concepts, lexicon, syntax, semantics) must be defined and shared before use. This means that the speed and scope of agent adaptability is constrained to the adaptibility of the human groups establishing the standards, and the human, sociotechnical standards propagation mechanisms. Also, these elements by definition cannot be directly responsive to situations in which language is used.

2. No existing standard ontology or classification scheme has ever actually proven to be stable in practice. New participants, interests, and representational elements emerge and must be accommodated in any ontology over time [3].

3. Even if shared concepts, ontologies, and linguistic elements were fixed and stable as representations, there is no way to guarantee stable interpretations of those representations in a fundamentally distributed context. Any interpretation depends on some interpretive procedure, which itself must be based on some standard to be shared, and the problem recurses for sharing and interpreting the interpretive procedure [5, 6].

4. Neither completely stable nor completely volatile ontological and linguistic structures are useful in any ultimate sense—a balance must be struck between the reliability of a conceptual or linguistic structure for joint action, and ability of such a structure to adapt to new representational and communicative needs. The degree of balance itself must be conditioned situationally.

Thus it's imperative to base multi-agent concept formation and communications on dynamic, distributed foundations. But which foundations? We need theories of the forms and limits of conceptual dynamics at the community level (and we are beginning to have these), and we also need specific implementation techniques that prescribe how to program individual agents so that their adaptive concept formation decisions lead to anticipatable global dynamics. Models of global concept dynamics need to be augmented by strong theories of individual and mutual agent concept formation that exhibit globally desirable properties. Ultimately, we are interested in formal and practical bases for programmable theories of dynamic linguistic behavior and ontology evolution in both individual agents and agent collections.

For these reasons, we are investigating some new machine learning techniques called *mutual online concept learning* (MOCL), that apply to multi-agent situations. In conventional machine learning, one agent serves as a learner and another as a teacher who provides instances. The learner's goal is to adapt to the teacher's concept, which is stable. In MOCL, no agent has a priveleged view, and every agent takes on both roles in an online fashion. There is no fixed global ("teacher's") referent for any concept since 1) every agent is both teacher and learner, 2) when teaching, each agent takes its own current version of a concept as the model of an instance to be transferred, and 3) when learning, each agent tries to adapt to its teacher's concept (so for n agents there are as many as n concepts being taught and learned). Since there is no obvious conceptual fixed point, the natural questions under this scenario include whether, when, how, and why multiple concepts can (fail to) converge to common concepts under different parameters. These are the primary questions investigated in this paper.

2. FRAMEWORK OF MUTUAL ONLINE CONCEPT LEARNING

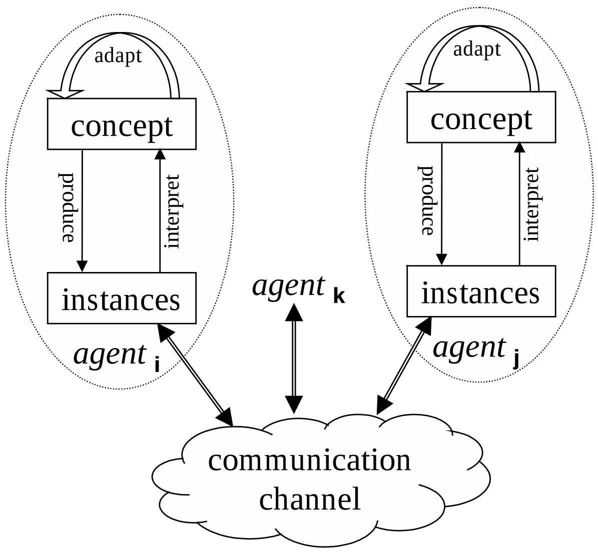

Figure 1: Framework of Mutual Online Concept Learning

A mutual online concept learning system consists of a set of agents and a communication channel used by agents to exchange information (see Fig. 1). In this paper we suppose the communication channel is perfect without noise. For one agent, the framework includes five elements: *concept*, *instances*, *instance producing mechanism*, *instance intrepreting mechanism*, and *concept adaption mechanism* (or *online concept learning*).

We explain the five elements as follows:

Concept
In broad sense, a concept can be thought of as a mapping from inputs to outputs. When treated as a *function* the output is deterministic for a given input. When treated as a mapping with associated probability distribution, the output is probabilistic. The functional (deterministic) point of view can be formalized as: $f : X \to Y$. For example, for Boolean concept, the domain $X = \{0,1\}^n$, and the range $Y = \{0,1\}$. When implementing a concept, we can use a variety of representations, such as propositions, first-order logic representations, or neural networks. The representation taken in this paper will be described in the following subsection.

Instance
When we view the concept as a function, the instance is just a specific case in the function's domain, such as $x \in X$.

Production
When an agent plans to express some concept to the outside world, it can use an inverse function to generate an instance to express this concept. There might exist many instances that can be used to express the concept if the mapping is not one-to-one.

Interpretation
Interpretation of instances is very easy under the functional interpretation of concepts: given an instance, apply the function to the instance and output the function value.

Adaptation
When an agent receives an instance from another agent, it may adapt its concept (e.g., probability, or network connection weights) such that the agent can perform a task better (e.g., gaining more benefits or payoff) next time with this new updated concept or knowledge. Formally speaking, suppose the concept of agent i at time t is $\mathcal{C}_i^t$, and the concept of agent j ($j \neq i$) at time t is $\mathcal{C}_j^t$. At time t, agent j generates an instance denoted by $instance(\mathcal{C}_j^t)$, and the instance was received by agent i. At next time $t+1$, agent i updates its concept to $\mathcal{C}_i^{t+1}$.

$$\mathcal{C}_i^{t+1} = adapt(\mathcal{C}_i^t,\ instance(\mathcal{C}_j^t)), \quad j \neq i$$

2.1 Comments

Representation
Representation is important and sometimes critical for system performance. In this paper, the instance space is an n-dimensional space $\mathbb{R}^n$ for real value or $\{0,1\}^n$ for Boolean value. The concepts to be learned from instances are *linear threshold concepts*, for which there is a hyperplane in $\mathbb{R}^n$ separating the points on which the function is 1 from the points on which it is 0 or -1.

Why do we take this kind of representation? First, linear threshold functions can be learned in online mode such as the Perceptron[18, 14, 4] and Winnow family algorithms[13]. Second, they have powerful representational capabilities. Third, there is much good theoretical work on this area, and many successful applications. For example, [19] showed that some commonly-used statistics-based and machine learning algorithms for natural language ambiguity resolution tasks can be re-cast as problems of learning linear threshold functions in the feature space.

Serial vs. concurrent interaction mode
Fig. 1 does not specify the time order of agent interaction. The interactions between agents can be classified into two modes: serial and concurrent. In the serial mode, at each time step only one agent is generating instances or updating

concepts; other agents are doing nothing.

In the concurrent mode, the two agents are simultaneously generating their instances and updating their concepts. There is no difference between the two agents in concurrent mode, while in the serial mode the two agents are not strictly equivalent since there must be one agent who starts the mutual process first, and so makes an initial impact on the direction of concept formation.

Single vs. multiple concepts
For convenience, the framework presented here only involves a single concept. However, the approach remains general. When we think of a concept as a function or a mapping, the common basis of multiple concept and single concept situations is evident, and there is much existing research on transforming "multi-class" (multiple-concept) problems to binary-class (single-concept) ones, e.g., [1].

Target concepts to be mutually learned
Before learning and interaction, initially, each agent may have its own individual target function or preferences. Given the multi-agent setting, the individual target function of different agents might be conflicting. They may want to colearn or collectively design a mutual target concept. In some sense, such a mutual concept can be thought as a balance, an equilibrium or an attractor. If every agent behaves under the mutual concept, they can get the (local) optimal benefit. If some agent moves away from a joint concept, it could be attracted back, or if a new agent joins the society it could break an existing balance (which may lead to a new, better result).

Closely related work: teacher-learner model
In [9], Goldman and Rosenschein proposed a similar teacher-learner model for mutually supervised learning, but their approach relied heavily on a specific traffic signal control scenario, which may be more difficult to generalize. It might be possible to transform their scenario into our more general representation based on n-dimensional space and linear threshold functions.

2.2 Two example scenarios

Emergence of language conventions
The above framework can be used to model the origins and evolution of language conventions such as a shared lexicon and grammar. In the case of lexicon conventions, the task of the agent community is to build a shared lexicon so that agents can understand each other[10, 21]. For example, let there be five symbols or sounds $\{a, o, e, i, u\}$, and some meanings in our mind or in the world. We represent each symbol as a feature that can take two values: $\{present, absent\}$. Therefore we have five Boolean features: $\{f_a, f_o, f_e, f_i, f_u\}$. Each point in the space can be used as a form to convey a meaning. Initially, different agents might use different forms to represent a given meaning. The purpose of the community is to form a shared form (e.g., $f_a \wedge f_i$) bound to a certain meaning through mutual adaptation.

Evolution of social conventions
The goal of agents may be to establish some social convention with some global optimal utility. This global optimum may be defined in several different ways, e.g. as the sum of individuals' utilities. There is significant work in this area, e.g., [20]. The strategy taken by an agent may be modeled as a concept instance, and it can also be represented as a point in an n-dimensional space. For example, let there be two agents $\{a, b\}$ and two strategies $\{s, t\}$. Then the simplest representation is using two features $\{f_a, f_b\}$, where each feature can take one of the two strategies. If history is considered, the feature space will of course be much larger. This kind of representation is widely used coding of strategies for "genetic" approaches, e.g, see [12]. Initially, an individual agent may not have any good idea (the concept to be sought) about what is an optimal strategy to take, but through mutual adaptation or evolution, agents may find joint strategies that form effective cooperative conventions. Note that in the game theory setting, the interaction mode is concurrent.

3. SINGLE-AGENT ONLINE CONCEPT LEARNING

We begin with an introduction to single-agent online concept learning, where the environment or teacher from which an agent learns is static. Then we discuss online concept learning in the multi-agent setting, i.e., *mutual online concept learning* (MOCL). In the MOCL setting, agents are acting as the environments of each other, and hence there is no "fixed" environment.

Online concept learning has also been called *online learning from examples* [13, 2]. The learning task is to discover the *target concept* or *target function* $f^*: \{0,1\}^n \to \{1, -1\}$ which maps each instance[1] to the correct *label* (or *class*) by responding to learning instances and feedback from a teacher.

The online learning takes place in a sequence of trials. In one trial, events occur in the following order:

1) A learner receives an instance from a teacher;
2) The learner labels the instance as 1 or -1;
3) The learner is told by the teacher whether or not the label was correct;
4) The learner uses the feedback to update its concept.

Each trial begins after the previous trial has ended.

3.1 Perceptron algorithm

The Perceptron algorithm was introduced first by Frank Rosenblatt [17] to solve the linear threshold learning problem. A preceptron (also called LTU – linear threshold unit) takes a vector of real-valued inputs[2], calculates a linear combination of these inputs, and outputs a 1 if the result is greater than some threshold and -1 otherwise. More precisely, given an input instance $\vec{x} = (x_1, ..., x_n)$, the output $f(\vec{x})$ computed by the perceptron is:

$$f(\vec{x}) = \begin{array}{ll} +1 & \text{if } \sum_{i=1}^{n} w_i x_i > \theta \\ -1 & \text{otherwise} \end{array} \quad (1)$$

where $\vec{w} = (w_1, ..., w_n) \in \mathbb{R}^n$, is the current weight vector maintained by the perceptron. For convenience, usually the threshold θ is set to 0. The reason is that we can add an additional constant input $x_0 = 1$ with a weight variable w_0.

For brevity, we will sometimes write the perceptron function as:

$$f(\vec{x}) = \mathrm{sgn}(\vec{w} \cdot \vec{x})$$

[1]We use *instance* and *example* interchangeably.
[2]The instance space $\{0,1\}^n$ is a special case of $\mathbb{R}^n$.

where

$$\text{sgn}(y) = \begin{array}{ll} +1 & \text{if} \quad y > 0 \\ -1 & \text{otherwise} \end{array}$$

Note that each weight vector defines a perceptron function or concept. Updating a weight vector is equivalent to updating a concept. So sometimes we may use *weight vector*, *function* or *concept* interchangeably.

Learning a concept involves choosing values for the weight vector $\vec{w} = (w_1, ..., w_n)$. Initially the algorithm starts with a weight vector $\vec{w} = (0, ..., 0)$. Upon receiving an instance $\vec{x} = (x_1, ..., x_n)$, the learner predicts the label of $\vec{x}$ to be $f(\vec{x})$. For brevity, sometimes we denote $f(\vec{x})$ by y.

If the predicted label is correct, then there are no changes in the weight vector. However, if the prediction is wrong, the weight vector is updated using the *perceptron learning rule*:

$$\vec{w} \leftarrow \vec{w} + \lambda y \cdot \vec{x} \tag{2}$$

where λ is the learning rate.

The Perceptron Convergence Theorem was proven in [15, 14, 4]; we sketch it here because we draw upon it in later proofs of mutual perceptron convergence.

THEOREM 1. **Perceptron Convergence Theorem** *If all instances are linearly separable, then a learner which uses the Perceptron algorithm will only make a finite number of mistakes. That is, the learning procedure converges.*

PROOF. The basic idea of the proof is to show that on each mistake made by the learner, the distance between the currently maintained weight vector (of the function f) and the target weight vector (of the target perceptron function f^*) becomes smaller after the update using the perceptron learning rule. □

4. MOCL: MUTUAL ONLINE CONCEPT LEARNING FOR MULTIPLE AGENTS

Multi-agent mutual concept learning is one of several possible extensions of single-agent online concept learning. In the multi-agent setting, each agent plays the roles of both teacher and learner. In this case, the environment that an agent learns from is dynamic since other agents, being part of the learner's environment, are also changing their concepts as they learn from each other. There is no fixed concept to be learned initially – no static teacher (or environment). The concept to be learned is dynamically formed as the result of the interactions among agents.

4.1 Mutual Concept Learning Game

We model multi-agent mutual concept learning as a *mutual concept learning game*. Initially each agent has its own concept. This concept can be generated using random weight vector. (In a language evolution context, this could be interpreted to mean that there is initially only a random relationship between meanings and communicative symbols.) Different agents might have different weight vectors at initialization. The game is played repeatedly in a (finite or infinite) sequence of rounds. On each round, the following events happen:

1. Two agents are randomly chosen from the agent population; one as learner and another as teacher. Denote by $\vec{w}_T$ the teacher's weight vector, and by $\vec{w}_L$ the learner's one.
2. The teacher randomly chooses a label from $\{1, -1\}$, and then generates an instance $\vec{x} = (x_1, ..., x_n)$ which is consistent with the label. An instance $\vec{x}$ is consistent with a label y_T, if and only if $\text{sgn}(\vec{w}_T \cdot \vec{x}) = y_T$.

   In terms of multiple concepts, randomly choosing a label from $\{1, -1\}$ can be understood as choosing a concept from a set of concepts.
3. The learner receives the instance $\vec{x}$, and predicts its label as: $y_L = \text{sgn}(\vec{w}_L \cdot \vec{x})$.
4. The teacher tells the learner whether its predicted label is correct or not.
5. The learner takes some actions (e.g., updating its weight vector or doing nothing), after getting the feedback from the teacher. The teacher also takes some actions (e.g., updating its weight vector or doing nothing) after sending the learner the feedback.

The goal of an agent in a learner's role is to make as few mistakes as possible during the mutual learning game. If possible, we would like all agents to converge to a shared concept so that no interpretation mistakes will be made in the population. (That is why we call the algorithm *mutual* learning.) We propose a mutual learning algorithm below which under some conditions will converge to a shared concept after making a finite number of mistakes.

4.2 The Mutual Perceptron Convergence Algorithm: A MOCL Algorithm

We know that the critical part of single-agent online concept learning is the actions taken by a learner after getting a teacher's feedback on its prediction. The approach taken by the Perceptron algorithm and the Winnow family of algorithms is to update the current concept only when a mistake is made. The difference between the Perceptron and Winnow algorithms is how weight vectors are updated.

In the multi-agent setting, the critical part is the same as in the single-agent setting. When a learner's prediction is correct, no action should be taken. The question is how to update weight vectors when the learner makes a mistake. The Mutual Perceptron Convergence Algorithm introduces an update rule that guarantees that the algorithm can converge under certain conditions.

Given two agents T as teacher and L as learner with their corresponding weight vectors: $\vec{w}_T$ and $\vec{w}_L$, then when updating weight vectors the *mutual perceptron learning rule* is:

$$\begin{array}{lcl} \vec{w}_L & \leftarrow & \vec{w}_L + \lambda y \cdot \vec{x} \\ \vec{w}_T & \leftarrow & \vec{w}_T - \lambda y \cdot \vec{x} \end{array} \tag{3}$$

Note that *both* learner and teacher will update their weight vectors. The intuition behind the joint update is that neither the teacher's nor the learner's existing concept is treated as the ideal reference concept. In a sense, the learner has made a mistake because the teacher, too, has made a mistake: if the teacher's concept had been correct with respect to the learner (i.e. identical to the learner's concept), then the learner would not have made a mistake.

4.3 Proof of Convergence

We want to show that under some conditions the Mutual Perceptron Convergence Algorithm will converge after making a finite number of mistakes. The basic idea is as follows.

If the learner makes a mistake on the instance given by the teacher, we want to show that the "distance" between the weight vectors of the two agents will become smaller after weight updating using the mutual perceptron learning rule.

However, if the number of agents is greater than two, we also need to consider the new "distances" between concepts of the learner/teacher agent and other agents in the population—we want the entire population to converge. We show that under some conditions, the *sum of the new distances* can be less or equal than the old distance, in which case the algorithm is guaranteed to globally converge.

Before going ahead to prove the convergence, we need some assumptions, some of which are also used in the classical single-agent perceptron convergence theorem proof.

Assumption 1. For any instance $\vec{x}$, there exists a positive constant κ such that $\|\vec{x}\| \leq \kappa$.

Assumption 2. For any weight vector $\vec{w}$ maintained by any agent, suppose for any instance and label pair $(\vec{x}, y)$ generated according to $y = \mathrm{sgn}(\vec{w} \cdot \vec{x})$, there exists a positive constant γ such that $y\vec{w} \cdot \vec{x} \geq \gamma > 0$.

To make the proof readable, we will first introduce some definitions as follows.

Definition 1. Given two agents with weight vectors $\vec{w}_A$ and $\vec{w}_B$, the *distance* between them is defined as:

$$\phi(\vec{w}_A, \vec{w}_B) = \|\vec{w}_A - \vec{w}_B\|^2$$

where $\|\cdot\|$ is the 2-norm (i.e., the Euclidean norm).

Definition 2. At time step t, the weight vectors of two agents A and B are $\vec{w}_A$ and $\vec{w}_B$. At time step $t+1$, the weight vectors become $\vec{w}'_A$ and $\vec{w}'_B$. The *variance of distance* between the two agents from time t to $t+1$ is defined as:

$$\varphi(\vec{w}_A, \vec{w}_B) = \phi(\vec{w}'_A, \vec{w}'_B) - \phi(\vec{w}_A, \vec{w}_B) = \Delta$$

Definition 3. Distance *reduction* is defined as $-\Delta$ if $\Delta < 0$.

Definition 4. Distance *introduction* is defined as Δ if $\Delta > 0$.

Lemma 1. Suppose on a round of the game, a learner L makes a mistake on the instance $\vec{x}$ given by a teacher T, and both agents use the mutual perceptron learning rule to update their weight vectors $\vec{w}_L$ and $\vec{w}_T$. Then there exists a positive constant $\delta = \frac{\gamma^2}{\kappa^2}$ such that:

$$\varphi(\vec{w}_T, \vec{w}_L) \leq -\delta. \tag{4}$$

when the learning rate $\lambda = \frac{\gamma}{2\kappa^2}$.

Proof. When the learner L makes a wrong prediction for the label of the instance $\vec{x}$ sent by the teacher T, it will modify its weight vector from $\vec{w}_L$ to $\vec{w}'_L$. According to the definition of label generation and prediction, we have:

$$\begin{array}{rcl} y & = & \mathrm{sgn}(\vec{w}_T \cdot \vec{x}) \\ -y & = & \mathrm{sgn}(\vec{w}_L \cdot \vec{x}) \end{array} \tag{5}$$

Then we have the following inequalities

$$\begin{array}{c} y \cdot \vec{w}_T \cdot \vec{x} > 0 \\ y \cdot \vec{w}_L \cdot \vec{x} < 0 \end{array} \tag{6}$$

With the above Assumptions 1 and 2, we have

$$\begin{array}{rcl} \varphi(\vec{w}_T, \vec{w}_L) & = & \phi(\vec{w}_T^{'}, \vec{w}_L^{'}) - \phi(\vec{w}_T, \vec{w}_L) \\ & = & \|(\vec{w}_T - \lambda y\vec{x}) - (\vec{w}_L + \lambda y\vec{x})\|^2 - \|\vec{w}_T - \vec{w}_L\|^2 \\ & = & -4\lambda y\vec{w}_T\vec{x} + 4\lambda y\vec{w}_L\vec{x} + 4\lambda^2\|\vec{x}\|^2 \\ & \leq & -4\lambda(\gamma - \lambda\kappa^2) \end{array}$$

When the learning rate $\lambda = \frac{\gamma}{2\kappa^2}$, we have

$$\varphi(\vec{w}_T, \vec{w}_L) \leq -\frac{\gamma^2}{\kappa^2}.$$

Let $\delta = \frac{\gamma^2}{\kappa^2}$, then $\varphi(\vec{w}_T, \vec{w}_L) \leq -\delta$ holds. □

The above lemma shows that two agents can move their concepts closer to each other through mutual learning or adaptation after making mistakes. However, for the situation of more than two agents, it is possible that moving one agent's concept toward that of another agent will cause it to move farther away from the concept of a third agent. Fig. 2 shows a very simple illustration.

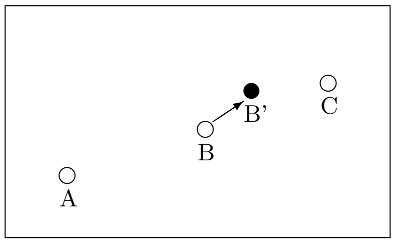

Figure 2: Agent B moves to B' which is closer to C, but the distance between B' and A becomes larger than before.

We want to show that under some conditions, the total distance *reduction* (see Definition 3) is larger than the total distance *introduction* (see Definition 4) after updating the weight vectors of the two choosen agents (teacher and learner) in a game round. The following lemma says that when the learning rate is set to the positive constant described in Lemma 1, then the *reduction* in the distance between the learner/teacher and other agents' concepts has a lower bound—there will be at least that much reduction in distance. Thus the point is that with such a learning rate, it is possible (but not yet certain) that the concepts may converge.

Lemma 2. Given an instance $\vec{x}$, and two agents A and B with weight vector $\vec{w}_A$ and $\vec{w}_B$. Suppose A's label y_A on $\vec{x}$ is different from B's label y_B, and A changes its weight vector $\vec{w}_A \leftarrow \vec{w}_A + y_B\vec{w}_A\vec{x}$, while B keeps its weight vector $\vec{w}_B$ unchanged. Let learning rate $\lambda = \frac{\gamma}{2\kappa^2}$, then there exists a constant $\Delta_r = 3\gamma^2/4\kappa^2 > 0$ such that

$$\varphi(\vec{w}_A, \vec{w}_B) \leq -\Delta_r.$$

Δ_r is called the *lower bound of distance reduction.*

Proof.

$$\begin{array}{rcl} \varphi(\vec{w}_A, \vec{w}_B) & = & \|(\vec{w}_A + \lambda y_B\vec{x}) - \vec{w}_B\|^2 - \|\vec{w}_A - \vec{w}_B\|^2 \\ & = & 2\lambda y_B\vec{w}_A\vec{x} - 2\lambda y_B\vec{w}_B\vec{x} + \lambda^2\|\vec{x}\|^2 \\ & \leq & -2\lambda\gamma + \lambda^2\kappa^2 \\ & = & -3\gamma^2/4\kappa^2 = -\Delta_r \end{array} \tag{7}$$

□

Assumption 3. For any weight vector $\vec{w}$ maintained by any agent, suppose for any instance and label pair $(\vec{x}, y)$ generated according to $y = \mathrm{sgn}(\vec{w} \cdot \vec{x})$, there exists a positive constant β such that $y\vec{w} \cdot \vec{x} \leq \beta$.

Next, Lemma 3 says that the distance introduced between the concepts of the learner/teacher and other agents has an upper bound.

Lemma 3. Given an instance $\vec{x}$, and two agents A and B with weight vector $\vec{w}_A$ and $\vec{w}_B$. Suppose A's label y_A on $\vec{x}$ is the same as B's label y_B, and A changes its weight vector $\vec{w}_A \leftarrow \vec{w}_A - y_B \vec{w}_A \vec{x}$ (note, this is a *mis-adaptation*), while B keeps its weight vector $\vec{w}_B$ unchanged. Let learning rate $\lambda = \frac{\gamma}{2\kappa^2}$, then there exists a constant $\Delta_i = \frac{\gamma}{\kappa^2}(\beta - \frac{3\gamma}{4}) > 0$ such that

$$\varphi(\vec{w}_A, \vec{w}_B) \leq \Delta_i.$$

Δ_i is called the *upper bound of distance introduction.*

Proof. It is obvious that $\Delta_i = \frac{\gamma}{\kappa^2}(\beta - \frac{3\gamma}{4}) > 0$ from the fact $\beta \geq \gamma$.

$$\begin{aligned} \varphi(\vec{w}_A, \vec{w}_B) &= \|(\vec{w}_A - \lambda y_B \vec{x}) - \vec{w}_B\|^2 - \|\vec{w}_A - \vec{w}_B\|^2 \\ &= 2\lambda y_B \vec{w}_A \vec{x} - 2\lambda y_B \vec{w}_B \vec{x} + \lambda^2 \|\vec{x}\|^2 \\ &\leq 2\lambda\beta - 2\lambda\gamma + \lambda^2\kappa^2 \\ &= \frac{\gamma}{\kappa^2}(\beta - \frac{3\gamma}{4}) = \Delta_i \end{aligned} \quad (8)$$

□

Now we are ready to give the proof of the mutual perceptron convergence theorem.

Theorem 2. *If $\frac{\beta}{\gamma} \leq \frac{3}{2}$ holds, then a population of agents which use the mutual perceptron algorithm can converge to a shared concept after making a finite number of mistakes.*

We call $\frac{\beta}{\gamma} \leq \frac{3}{2}$ as the *mutual perceptron convergence condition.*

Proof. The idea is simple. After a learner makes a mistake, we hope that by using our mutual perceptron learning rule for each weight vector update, the sum of new inter-concept distances among all agents is smaller than before. Suppose there are N agents. Among these N agents, there are two agents, learner and teacher respectively, whose corresponding weight vectors are $\vec{w}_L, \vec{w}_T$. The weight vectors of other agents are $\vec{w}_1, ..., \vec{w}_{N-2}$. (to be consistent, let $\vec{w}_L = \vec{w}_{N-1}$, and $\vec{w}_T = \vec{w}_N$.) Then the sum of distances can be written as:

$$\begin{aligned} \sum_{i,j=1}^N \phi(\vec{w}_i, \vec{w}_j) &= \phi(\vec{w}_T, \vec{w}_L) + \sum_{i,j=1}^{N-2} \phi(\vec{w}_i, \vec{w}_j) \\ &+ \sum_{i=1}^{N-2} \phi(\vec{w}_T, \vec{w}_i) + \sum_{i=1}^{N-2} \phi(\vec{w}_L, \vec{w}_i) \end{aligned} \quad (9)$$

And the sum of new distances after the mutual learning is written as:

$$\begin{aligned} \sum_{i,j=1}^N \phi^{'}(\vec{w}_i, \vec{w}_j) &= \phi^{'}(\vec{w}_T, \vec{w}_L) + \sum_{i,j=1}^{N-2} \phi^{'}(\vec{w}_i, \vec{w}_j) \\ &+ \sum_{i=1}^{N-2} \phi^{'}(\vec{w}_T, \vec{w}_i) + \sum_{i=1}^{N-2} \phi^{'}(\vec{w}_L, \vec{w}_i) \end{aligned} \quad (10)$$

where $\phi^{'}(\vec{w}_i, \vec{w}_j)$ is the brief notation of $\phi^{(}\vec{w}_i^{\prime}, \vec{w}_j^{\prime})$.
Now let us compute the new total distances. Let $\vec{x}$ be the instance on which the learner made a mistake. Denote by y_L the label computed by the learner, and by y_T the label from the teacher. On the instance $\vec{x}$, other $N-2$ agents also have their own labels, denoted by $y_1, ..., y_{N-2}$. Among these $N-2$ agents, suppose there are p agents whose labels are the same as y_T, and q agents whose labels are the same as y_L, where $p+q = N-2$. Without loss of generality, suppose $y_1 = \cdots = y_p = y_T$ and $y_{p+1} = \cdots = y_{p+q} = y_L$.

So, according to the Lemma 2 and 3, the total distance variance between the teacher's concept and those of the other agents (excluding the learner) is at most $-p\Delta_r + q\Delta_i$. This is because we have

$$\begin{aligned} \sum_{i=1}^{N-2} \phi^{'}(\vec{w}_T, \vec{w}_i) &= \sum_{i=1}^{p} \phi^{'}(\vec{w}_T, vwi) + \sum_{i=p+1}^{N-2} \phi^{'}(\vec{w}_T, \vec{w}_i) \\ &\leq \sum_{i=1}^{p} \phi(\vec{w}_T, \vec{w}_i) - p \cdot \Delta_r \\ &+ \sum_{i=p+1}^{N-2} \phi(\vec{w}_T, \vec{w}_i) + q \cdot \Delta_i \end{aligned}$$

and similarly, the total distance variance between the learner's concept and those of the other agents (excluding the teacher) also is at most $p\Delta_i - q\Delta_r$. Again, this is because we have:

$$\begin{aligned} \sum_{i=1}^{N-2} \phi^{'}(\vec{w}_L, \vec{w}_i) &= \sum_{i=1}^{p} \phi^{'}(\vec{w}_L, \vec{w}_i) + \sum_{i=p+1}^{N-2} \phi^{'}(\vec{w}_L, \vec{w}_i) \\ &\leq \sum_{i=1}^{p} \phi(\vec{w}_L, \vec{w}_i) + p \cdot \Delta_i \\ &+ \sum_{i=p+1}^{N-2} \phi(\vec{w}_L, \vec{w}_i) - q \cdot \Delta_r \end{aligned}$$

Putting these two inequalities together, plus

$$\sum_{i,j=1}^{N-2} \phi^{'}(\vec{w}_i, \vec{w}_j) = \sum_{i,j=1}^{N-2} \phi(\vec{w}_i, \vec{w}_j)$$

and Eq.(4):

$$\phi^{'}(\vec{w}_T, \vec{w}_L) \leq \phi(\vec{w}_T, \vec{w}_L) - \delta,$$

we can express the total distance between all agents as follows (following Eqs. (9) and (10)):

$$\sum_{i,j=1}^{N} \phi^{'}(\vec{w}_i, \vec{w}_j) \leq \sum_{i,j=1}^{N} \phi(\vec{w}_i, \vec{w}_j) + (p+q) \cdot (\Delta_i - \Delta_r) - \delta$$

If the mutual convergence condition $\frac{\beta}{\gamma} \leq \frac{3}{2}$ holds, then the amount of *reduction* in distance is equal or greater than the *introduction* in distance, i.e., the difference is less than or equal to zero:

$$\begin{aligned} \Delta_i - \Delta_r &= \tfrac{\gamma}{\kappa^2}(\beta - \tfrac{3\gamma}{4}) - \tfrac{3\gamma^2}{4\kappa^2} \\ &= \tfrac{\gamma}{\kappa^2}(\beta - \tfrac{3}{2}\gamma) \\ &\leq 0 \end{aligned}$$

Therefore, the sum of new inter-concept distances among all agents *is* smaller than before:

$$\sum_{i,j=1}^{N} \phi^{'}(\vec{w}_i, \vec{w}_j) \leq \sum_{i,j=1}^{N} \phi(\vec{w}_i, \vec{w}_j) - \delta$$

where $\delta = \frac{\gamma^2}{\kappa^2}$.
Suppose the initial total distances among all agents is Δ_σ, then the algorithm will converge after making at most $\frac{\kappa^2}{\gamma^2}\Delta_\sigma$ mistakes. □

4.4 Analysis of the mutual perceptron convergence condition

What is the interpretation of $\frac{\beta}{\gamma} \leq \frac{3}{2}$?

We want to show that there is a relationship between the quality of the concept instance produced by the teacher, and the possibility of convergence. ¿From Assumptions 2 and 3 we know that,

$$\gamma = \min_{y,\vec{w},\vec{x}} \|y\vec{w}\vec{x}\| \quad (11)$$

$$\beta = \max_{y,\vec{w},\vec{x}} \|y\vec{w}\vec{x}\| \quad (12)$$

Without loss of generality, suppose all instances have unit length. So, rewriting, we have:

$$\|y\vec{w}\vec{x}\| = |y| \cdot \|\vec{w}\| \cdot \|\vec{x}\| \cdot |cos(\vec{w},\vec{x})| = \|\vec{w}\| \cdot |cos(\vec{w},\vec{x})|$$

Then, the ratio between γ and β can be rewritten as:

$$\frac{\gamma}{\beta} = \frac{\min_{\vec{w},\vec{x}} \ \|\vec{w}\| \ |\cos(\vec{w},\vec{x})|}{\max_{\vec{w},\vec{x}} \ \|\vec{w}\| \ |\cos(\vec{w},\vec{x})|}$$

Suppose max $|\cos(\vec{w},\vec{x})| = 1$
and

$$\max \|\vec{w}\| = \rho \ \min \|\vec{w}\|$$

where $\rho \geq 1$.

Now we have

$$\gamma/\beta = \frac{1}{\rho} \min_{\vec{w},\vec{x}} \ |\cos(\vec{w},\vec{x})|$$

Given all above assumptions, $\frac{\beta}{\gamma} \leq \frac{3}{2}$ is equivalent to:

$$\min_{\vec{w},\vec{x}} \ |\cos(\vec{w},\vec{x})| \geq \frac{2}{3}\rho$$

where $1 \leq \rho \leq \frac{3}{2}$.

This analysis means that a sufficient condition for convergence is that the instance generated by a teacher falls within a certain distance region within the space of all possible instances. This idea is expressed visually in Fig. 3.

The *most* representative instance, represented by the dark vector in Fig. 3, occurs when $\vec{x} = \eta\vec{w}$, where $\eta \in \mathbb{R}$ and $\eta \neq 0$, that is, when the instance coincides with the weight vector or negative weight vector exactly.

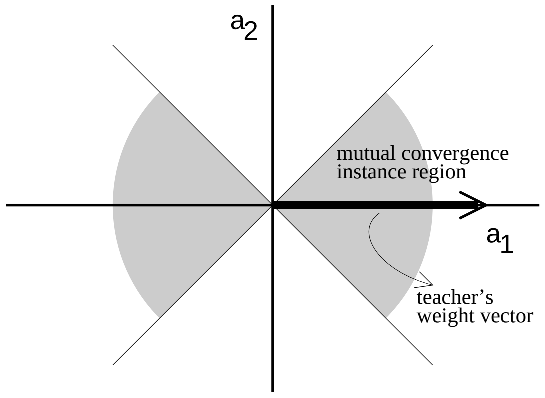

Figure 3: Region of instances satisfying the mutual convergence condition ($\rho = 1.06$)

The shaded area of Fig. 3 shows the region of instances that, when generated by the teacher, will satisfy the mutual convergence condition given $\rho = \frac{3\sqrt{2}}{4} \approx 1.06$. Happily, this region occupies half of the entire instance space.

4.5 Discussion

There are several additional points to make about convergence. First, even though several pairs of agents may simultantously updating their concepts independently in local serial games, convergence is not affected. This is because the total distance reduction (as computed in the proof of Theorem 2) doesn't change.

Second, in theory, it seems that the convergence condition might be very hard to satisfy. But in practive, convergence is easy to obtain, even when some of the strict convergence conditions (such as the values of γ and β) are violated. We have studied this in simulations, results of one of which are shown in Fig. 4. Mistakes are shown for each generation. In each generation, each agent has 10 chances of interacting with other agents. That means one generation involves 100 rounds of the game if there are 10 agents, which also indicates that the number of mistakes in one generation will not be larger than 100.

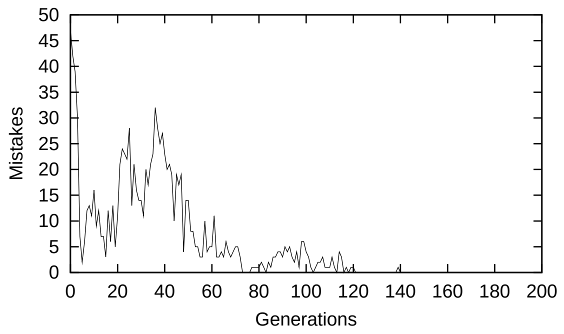

Figure 4: Simulation on the convergence of ten agents with $\beta/\gamma = 9.6/0.2 = 48 \gg 3/2$**.**

This figure shows convergence of ten agents (instance space is $\{0,1\}^{10}$) to a mutual concept after about 140 generations, with $\gamma = 0.2$ and $\beta = 9.6$. These two values are obtained during the simulation according to their definitions in Assumptions 2 and 3.

To give readers some image of what the concepts look like before and after convergence, we conducted a simulation on two agents on an instance space $\{0,1\}^3$ with three Boolean variables $\{x_1, x_2, x_3\}$ and a threshold $\theta = 3$. The two initial concepts are x_1 and x_2, and the converged concept is $x_1 \wedge x_2$. The weight vectors are shown in Table 1.

Table 1: Weight vectors of concepts before and after convergence.

	concepts	weight vector			$\theta = 3$
		w_1	w_2	w_3	
initial	x_1	3.1	1.2	0.7	$\rightarrow$ only $w_1 > \theta$
	x_2	0.5	3.3	1.3	$\rightarrow$ only $w_2 > \theta$
converged	$x_1 \wedge x_2$	1.6	2.1	0.5	$\rightarrow$ only $w_1 + w_2 > \theta$

5. CONCLUSION AND FUTURE WORK

As one of our efforts towards constructing the basis for programmable theories of dynamic communication behavior in both individual agents and agent collections, we have proposed the mutual online concept learning framework. This framework views the knowledge/language of a multi-agent system as a dynamic set of concepts, where each agent has its own concept/instance processing (production, interpretation, and adaptation) mechanism. In this framework, each agent uses its current version of a concept to produce and publish an instance, and other agents can use that instance update their own concepts. There is no pre-existing global concept to be learned; instead, agents are in effect collectively designing a concept that is evolving as they exchange information.

We think the Mutual Online Concept Learning framework is a general one that can be used for the study of emergence and evolution of many kinds of information structures including language, ontologies, and subject indexes. When the concept used here is extended to multiple categories to represent word meaning, and the instances are used to represent word form (especially for constructing structured word forms of the sort used by people), MOCL provides a natural approach to an algorithm for vocabulary convergence. We also believe that if we were to extend the concept to multiple categories represented as a vector (points in a multi-dimensional space), we might also get a grammatical structure convergence algorithm. These extensions remain for future work.

6. ACKNOWLEDGMENTS

This work was partially supported under NSF grant 9981-797, and by the Information Systems Research Lab (ISRL) of the Graduate School of Library and Information Science. We thank Vic Lesser, his students, and the reviewers for helpful comments, and Bei Yu and Terry von Thaden for their encouragement and support of this research.

7. REFERENCES

[1] E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *Proc. 17th International Conf. on Machine Learning*, pages 9–16. Morgan Kaufmann, San Francisco, CA, 2000.

[2] A. Blum. On-line algorithms in machine learning. In *Proceedings of the Workshop on On-Line Algorithms*, Dagstuhl, 1996.

[3] G. C. Bowker and S. L. Star. *Sorting Things Out: Classification and Its Consequences.* MIT Press, Cambridge, MA, 1999.

[4] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, 1973.

[5] L. Gasser. Social conceptions of knowledge and action: Distributed artificial intelligence and open systems semantics. *Artificial Intelligence*, 47:107–138, 1991.

[6] L. Gasser. Boundaries, identity and aggregation: Plurality issues in multi-agent systems. In Y. Demazeau and E. Werner, editors, *Decentralized Artificial Intelligence III.* Elsevier, 1992.

[7] M. R. Genesereth. Knowledge interchange format, 1998. draft proposed American National Standard (dpANS), NCITS.T2/98-004.

[8] P. Gmytrasiewicz and M. Huhns. The emergence of language among autonomous agents. *IEEE Internet Computing*, 4(4):90–92, 2000.

[9] C. V. Goldman and J. S. Rosenschein. Mutually supervised learning in multiagent systems. In G. Weiß and S. Sen, editors, *Adaptation and Learning in Multi–Agent Systems*, pages 85–96. Springer-Verlag, 1996.

[10] N. Komarova and M. Nowak. The evolutionary dynamics of the lexical matrix. *Bull. Math. Biol*, 63(3):451–485, 2001.

[11] Y. Labrou and T. W. Finin. Semantics and conversations for an agent communication language. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 235–242. Nagoya, Japan, 1997.

[12] K. Lindgren. Evolutionary phenomena in simple dynamics. In C. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pages 295–312. Addison-Wesley, 1992.

[13] N. Littlestone. Learning quickly when irrelevant attributes abound: a new linear threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

[14] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, 1969.

[15] A. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata, volume XII*, pages 615–622, 1962.

[16] C. Reed, T. Norman, and N. Jennings. Negotiating the semantics of agent communication languages. *Computational Intelligence*, 2002.

[17] F. Rosenblatt. The perceptron: A probabilitic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958.

[18] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Spartan Books, New York, 1962.

[19] D. Roth. Learning to resolve natural language ambiguities: a unified approach. In *Proceedings of 15th Conference of the American Association for Artificial Intelligence*, pages 806–813, Madison, US, 1998.

[20] Y. Shoham and M. Tennenholtz. Co-learning and the evolution of social activity. Technical report, Computer Science Department, Stanford University, 1993.

[21] L. Steels. Self-organizing vocabularies. In C. Langton and T. Shimohara, editors, *Artificial Life V*, Nara, Japan, 1996.

[22] L. Steels. The origins of ontologies and communication conventions in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 1(2):169–194, October 1998.