The Classification Game: Combining Supervised Learning and Language Evolution

Samarth Swarup
Virginia Bioinformatics Institute,
Virginia Polytechnic Institute and State University,
Blacksburg, VA 24061,
swarup@vbi.vt.edu

Les Gasser
Graduate School for Library and Information Science, and
Department of Computer Science,
University of Illinois at Urbana-Champaign,
Champaign, IL 61820,
gasser@illinois.edu
Abstract

We study the emergence of shared representations in a population of agents engaged in a supervised classification task, using a model called the Classification Game. We connect languages with tasks by treating the agents’ classification hypothesis space as an information channel. We show that by learning through the classification game, agents can implicitly perform complexity regularization, which improves generalization. Improved generalization also means that the languages that emerge are well-adapted to the given task. The improved language-task fit springs from the interplay of two opposing forces: the dynamics of collective learning impose a preference for simple representations, while the intricacy of the classification task imposes a pressure towards representations that are more complex. The push-pull of these two forces results in the emergence of a shared representation that is simple but not too simple. Our agents use artificial neural networks to solve the classification tasks they face, and a simple counting algorithm to learn a language as a form-meaning mapping. We present several experiments to demonstrate that both compositional and holistic languages can emerge in our system. We also demonstrate that the agents avoid overfitting on noisy data, and can learn some very difficult tasks through interaction, which they are unable to learn individually. Further, when the agents use simple recurrent networks to solve temporal classification tasks, we see the emergence of a rudimentary grammar, which does not have to be explicitly learned.

1 Introduction

It is very hard to imagine what life would be like without language — nearly everything we do depends on it. Language structures the way we think and act. It helps us to plan ahead, and to adapt collectively to changing circumstances. Language itself changes over time, and changes in language reflect both our learning biases and changes in our environment — what we need to talk about. Understanding the processes that lead to language change, and their interaction with cultural and biological dynamics, is an important scientific problem.

Language evolution is important from a technological viewpoint also. As we are learning to build increasingly complex and sophisticated machines, we are finding the need to endow them with increasing degrees of autonomy so that they can learn, plan, and operate on their own. Added autonomy is necessary because not every eventuality can be foreseen and not every required response can be programmed in. While we have managed to create many degrees of autonomy for intelligent machines, so far we do not know how to enable machines to develop and adapt their own languages. We believe this is an essential step in the creation of autonomous populations. Language is, in many ways, the last frontier of autonomy (Gasser, 2007).

In order to achieve these goals, we need to understand how language can emerge and evolve. In the last several decades, the study of artificial language evolution has been gaining momentum. Partly this is because language is unique. If there were other species with language, comparative
analysis could help us develop a theory of language emergence and change. This is still possible to an extent by studying animal communication systems (Hockett, 1960; Swarup & Gasser, 2007), but those don’t exhibit the complexity and richness of human language. To a large extent, any explanation of the emergence of human language is going to be a “just so” story.

However, when we include artificial language evolution in the mix, we can ask more general questions such as what are the minimal initial conditions for the emergence of language? In other words, we can hope to uncover some general principles underlying the emergence and change of language by experimentation with artificial agents and idealized mathematical models. One of the major directions of research in this respect has been the use of “language games” (see Vogt (2006) for a review), starting with the introduction of the Naming Game by Steels (1996).

In the Naming Game, a pair of agents is selected randomly from a population, and assigned the roles of speaker and hearer. These agents are presented with a scene (called “context”) containing a few objects. The speaker selects one object to name, and the hearer has to interpret the speaker’s utterance and point correctly to the chosen object. Updates to the agents’ languages can be made in two ways. In the first case the hearer is told which object was chosen by the speaker, and thus the hearer can calculate an error measure. This allows the hearer to update its own language so as to directly reduce the error — for example by following the error gradient. In the second case, agents only get to know whether the communication was successful (this is like a simple reinforcement signal). In the case of failure, the hearer will not know the correct referent for the utterance and will have to rely on contextual information, such as seeing multiple instances of the object in different situations, to arrive at the correct mapping from utterances to objects. Steels showed that repeating this process with random pairs of agents eventually results in the emergence of a shared lexicon in the population.

In addition, the agents can also be required to learn to discriminate the objects from one another. The features they learn to pick out in this process can be thought of as the “meanings” with which symbols become associated.

A few different variations of this basic language game have been studied. Vogt and Coumans (2003) group these variants into three categories: the Guessing Game, in which the hearer has to guess the object chosen by the speaker and is given the right answer afterward (described above), the Observational Game, where joint attention is established (by pointing to an object, for example) before communication is attempted, and the Selfish Game, where there is no feedback about the success of communication. In this last case, the agents have to rely on seeing sufficiently many contexts to arrive at a shared language. For this reason, the Selfish Game is also known as the Cross-Situational Learning Game (Smith, Smith, Blythe, & Vogt, 2006). The main difference Vogt and Coumans note between the three games is that the Selfish Game results in a much slower convergence than the other two, which are indistinguishable in terms of convergence time.

In all of these games, the primary goal for agents is creating a shared language. Meanings are created through the process of the language game. Consequently meanings have no purpose
The Classification Game

other than validating communication itself, and no relevance to any action or task. In order to
go beyond this somewhat circular paradigm, we need to be able to connect language to tasks, so
that the emergent language is for something. In this article, we introduce the Classification Game,
which connects language emergence with classification tasks. Classification tasks are a very general
category of tasks, and they have the additional advantage of being well-studied in machine learning.
The Classification Game protocol is similar to the protocols described above. The speaker and
hearer are presented with an example from a training set to classify. The speaker has to generate an
utterance that tells the hearer how to classify the point. The hearer ignores the example itself, and
generates the label by decoding the speaker’s utterance. Both speaker and hearer are then given
the correct label for the example and both can update their classifiers and languages. We will show
below how this setup is based on an analogy between an information channel and the hypothesis
space being used by the agents. This will allow us to establish a clear relationship between the
language and the task, and to investigate the optimality of the emergent languages.

The rest of this article is organized as follows. First we discuss two major selection pressures
in language evolution: learnability and functionality. (We will show in later sections that the clas-
sification game sets up a trade-off between these two pressures that results in languages that are
well-adapted to the task at hand.) We then describe how learning can be viewed as a communication
problem and use this to set up the description of the classification game. We follow this with
a series of experiments to demonstrate the pressures towards learnability and functionality, and to
examine the kinds of languages that can emerge, which include both holistic and compositional
kinds. We also demonstrate that the “adaptedness” of language to task is equivalent to complexity
regularization in learning, which helps to achieve better generalization by avoiding overfitting. The
benefit of interaction is further demonstrated by a gender-classification experiment in which agents
using vanilla neural networks (and no communication) are unable to achieve better than random
accuracy, but the use of the interactive classification game enables agents to do better. We then use
recurrent neural networks to demonstrate the emergence of more structured language when the task
is temporal in nature. We end with a discussion of related work, and of directions for future work.

2 Selection pressures in language evolution

In recent years, the application of the evolutionary metaphor to language change has gained cur-
rency. A natural question this raises is, what determines “fitness” of a language? Linguists often
answer by attributing extrinsic sources of fitness, such as the prestige of the speaker (Croft, 2001;
Mufwene, 2002).

For human languages it is generally accepted that intrinsic factors such as the learnability of a
language do not vary across languages. A child born into a Hindi-speaking community will learn
Hindi as easily as a child born into an English-speaking community will learn English. Modern
languages, however, have adapted over a long period of time. If we go far enough into the history of language, it is clear that (aspects of) early languages had differential fitness. For example, Phoenicians were the first to develop a phonetic alphabet. This innovation quickly became established in many languages, even though the Phoenician language itself died out. One explanation is that phonetic alphabets fixated because a phonetic writing system is much easier to learn.

Earlier experiments have shown us that there is a pressure for learnability in collective language learning. In earlier work, we trained a population of simple recurrent neural networks to come to consensus on a language (Swarup & Gasser, 2006). We saw that the population tended to converge upon a maximally simple “language”, such as always generating a single symbol over and over. This suggests that simple mappings spread easily through a population, probably because they can be learned more quickly. From the point of view of the evolution of representations, learnability provided an *intrinsic fitness* to the languages. The immediate question this raises, then, is how could complex languages ever evolve if the only fitness component for language were simplicity? To allow the emergence of more complex and thus more useful languages in simulation, a bias for increasing complexity has to be built in by the experimenter (Briscoe, 2003). Where would this bias come from, if it exists, in a natural system?

It seems intuitive that the counter-pressure for making language more complex must come from the *functionality* of language. A language is for something. In other words, agents gain some benefit from having a particular language/representation. If the use of a particular language gives an agent high reward (perhaps through low error on some task), then the agent should try to maintain that language. This reward then gets transferred to the language as a boost in fitness. Languages that are too simple, however, are unlikely to be very functional, because their information-carrying capacity is low. Hence an agent should feel a pressure to discard such a language.

Thus we can imagine that languages occupy a complexity line, with complexity increasing to the right, as shown in figure 1. Learnability increases with simplicity, i.e., to the left, and expressiveness or functionality increases with complexity, i.e., to the right. There are several ways of measuring the complexity of a language. We could, for example, try to compress a corpus of utterances using a standard compression algorithm. A greater degree of compression would imply lower (Kolmogorov) complexity of the language. Alternatively, we could evaluate the complexity of acquisition, which is the number of examples required for learning the language (in a Probably Approximately Correct, or PAC, setting, for example). Increasing complexity would correspond to increasing numbers of examples required for acquisition. Yet another method is to assume some
distribution (say, uniform) over meanings, and then evaluate the entropy of the resulting set of utterances. In this case higher entropy implies higher complexity.

In general, for a given number of meanings to be expressed, there will be more than one language capable of expressing them. Importantly, these languages will not all have the same complexity. Suppose we define the complexity of a language to be its information entropy. Then a holistic language, which assigns a unique utterance to each meaning, will have higher complexity than a compositional language, which assigns unique utterances to a subset of the meanings and expresses the rest using combinations of these basic utterances. If we assume a uniform distribution over meanings, it is easy to see that the set of utterances generated by the holistic language will have the highest entropy. The set of utterances generated by the compositional language will have lower entropy because of repetitions. Conversely, for a given complexity (entropy) value, a compositional language will be more expressive than a holistic language.

At the other end, zero complexity languages are the ones that are referentially useless. They correspond to saying nothing at all, or saying the same thing at every instant. In other words, they carry no information.

Ideally we would like the languages that evolve to be in the region that is “just right”, i.e., where the language that evolves is both easily learnable and useful. Such a language would be well-adapted to the task at hand.

The goal of this article is to relate language to task in a way that allows a population of agents to jointly learn a shared representation that is well-adapted to the complexity of the task. We do this by exploiting the Classification Game described above, in which agents interact with each other while learning to perform a classification task. The interaction between agents results in the emergence of a shared representation—a language—that is simple, but not too simple. As we have noted, the Classification Game is based on a communication view of supervised learning, which we describe next, before describing the interaction protocol and the learning algorithms.

3 Combining supervised learning and communication

We suppose that the agents are learning to perform some two-class classification task. This means that the agents are given a set of input examples, and are told the class labels for these examples. We assume that the examples are divided into just two classes (positive and negative, say). The agents search through a hypothesis space, \( \mathcal{H} \), to find the best classifier. This learning process is illustrated in fig. 2. The hypothesis space shown in the figure consists of straight lines, or more generally, hyperplanes, of varying positions and orientations. The learning task is to find a hyperplane which maps the inputs onto the appropriate labels. If such a hyperplane exists, we say that the inputs are linearly separable. There may be many such hyperplanes, or there might be none, depending on the arrangement of the input points. If there is no hyperplane that can completely separate the two
classes of points, it is still be possible to find a hyperplane that minimizes the error, i.e. misclassifies the fewest points.

Fig. 3 shows the standard (Shannon) view of a communication process. The sender wants to transmit some information over a noisy channel to a receiver. The sender encodes his message into a channel code using his encoder function. At the other end of the channel, the receiver uses a symmetric decoder to retrieve the message. Since the channel is noisy, some of the bits of the encoded message might be distorted (changed) during the passage from the sender to the receiver. In this case, the sender can choose a longer encoding to ensure that the receiver gets the right message. In fact, if the channel is not completely random, it is possible to compensate for any amount of noise by using a sufficiently long encoding of the messages. This is the fundamental theorem of information theory, also known as the noisy-channel coding theorem (Shannon, 1949).

We relate learning to communication by treating $\mathcal{H}$ as an information channel. This is illustrated in Fig. 4. Note that, in this analogy, the sender and receiver are both components of a single agent’s “cognitive architecture”. Learning is conceived of as a process wherein the sender encodes the input examples using one or more hypotheses from the hypothesis space, and the receiver decodes them into the labels for these examples.

This perspective is explained through a specific example: the xor problem. In this problem, we have four points, divided into two classes. The points are described by using two bits:
The Classification Game

Figure 4: The learning-communication analogy. The hypothesis space is equivalent to the channel. The learner has to encode the inputs in such a way that they can be decoded into the appropriate labels. Note that, in this analogy, the “sender” and the “receiver” are both components of a single agent.
{(0, 0), (0, 1), (1, 0), (1, 1)}. A point is labeled positive (or 1) if the two bits describing it are different, otherwise it is labeled negative (or 0). Mathematically, the label is the Boolean \textit{xor} of the input bits. Thus, in sequence, the labels for the points are 0, 1, 1, 0 (or just 0110).

If the hypothesis space available to an agent to solve this classification problem is that of straight lines, then at least two hypotheses are required to separate the positive and negative points, as shown in fig. 5. There is no single straight line which can separate the positive and negative points. In terms of the information channel analogy, the message 0110 is “forbidden”. In fact, it can easily be seen that all other messages (i.e., sequences of four labels) are achievable, except 1001 (which is simply the inverse of 0110).

This corresponds to the case where the hypothesis space is equivalent to a noisy channel. Since it only forbids two out of the sixteen possible messages, it should be possible to transmit all messages by using a longer encoding, as stated by the noisy-channel coding theorem. One way to solve the encoding and decoding problem is as shown in fig. 5, using two straight lines oriented as shown.

Proceeding in sequence from (0, 0) in figure 5, the encodings generated by hypothesis A are,

{(0, 0) → 0,
(0, 1) → 1,
(1, 0) → 1,
(1, 1) → 1,}

and those by hypothesis B are,

{(0, 0) → 1,
(0, 1) → 1,
(1, 0) → 1,
(1, 1) → 0,}

Combining the two, each point gets a two bit encoding: 01, 11, 11, and 10 respectively. The decoder function converts these into labels by using another hypothesis from the hypothesis space.
The Classification Game

as follows:
11 → 1,
10 → 0,
01 → 0,

Thus we can classify all the points correctly. The learning problem, thus, is to determine an encoding and decoding function from a given hypothesis space, for a given set of input examples and labels (known as a training set).

In the xor problem above, we were given all the points. In general this is not the case; the learning problem has to be solved from a subset of the possible points. Thus one of the key concerns is generalization, i.e. being able to label correctly points that are not present in the training set. It can be shown that the expected generalization error is lower for more compact encodings (Blumer, Ehrenfeucht, Haussler, & Warmuth, 1987; Board & Pitt, 1990; Li, Tromp, & Vitányi, 2003). Thus it is important, while searching through the hypothesis space for encoding and decoding functions, to try to find functions of low complexity. There are several techniques that have been proposed to do this (Barron, 1991; Hinton & Camp, 1993; Lugosi & Zeger, 1996; Hochreiter & Schmidhuber, 1997; Kärkkäinen & Heikkola, 2004). We will see, in what follows, that the Classification Game can solve this problem implicitly.

4 The Classification Game

Now we describe the experimental setup in which agents interact with each other and learn to solve a classification problem while also learning to communicate about it. The essential idea is that, in an interaction, one of the agents communicates its encoding of the given example not only to its own decoder, but also to the decoder of the other agent. This is illustrated in fig. 6. Thus, over the course of repeated interactions, the population of agents have to converge upon the same encoding and decoding functions for the given classification problem.

We choose hyperplanes as our hypothesis class. This leads very naturally to artificial neural networks as the implementation. Each hidden layer node of an artificial neural network, called a perceptron, can be thought of as a hyperplane. The number of hidden layer nodes in each agents’ neural network defines the number of hyperplanes it can potentially use to classify the training examples. The first, or hidden layer, is the encoder. The second, or output layer, has just a single node. This is the decoder.

To exchange information, the agents convert the outputs of their encoders into a public language using a form-meaning mapping. This is a matrix $[f_{ij}]$ in which each entry defines the likelihood of pairing form $i$ (typically a letter of the alphabet), with meaning $j$ (a hidden layer node number). In other words, encodings are not directly communicated between speakers and hearers. In addition to developing correct classification hypotheses, agents have to learn how to map the set of public
symbols to their internal representations.

The protocol is simple. At each step, we select two agents uniformly randomly from the population. One agent is assigned the role of speaker, and the other is assigned the role of hearer. Both are presented with the same training example. The speaker uses the outputs of its encoder to generate an utterance describing its hypothesis (hidden layer node activations) in the public language using its form-meaning mapping. The hearer tries to decode this utterance via its own form-meaning mapping, and uses its decoder to generate a label based on this decoding. Both agents are then given the expected label, whereupon they can update their neural networks and form-meaning mappings. This entire process is illustrated in figure 7.

The use of the form-meaning mapping is best explained through an example. The form-meaning mapping matrix is initialized to zeroes for all the agents. Suppose that the speaker has four hidden layer nodes, and that on the current example, the speaker’s hidden layer activations are (0.6, 0.4, 0.1, 0.9). Nodes with activations above 0.5 are considered “active”. Thus the nodes that are active for the speaker are the first and the fourth. The speaker will examine the first column of its form-meaning mapping to find the first entry with the highest value. Since all entries are zeroes at this point, the entry in the first row is the first-highest value, and it corresponds to the symbol A. Agents update their form-meaning matrices both while speaking and while hearing, so the speaker will now update the entry in location (1, 1) of its form-meaning matrix by adding a strengthening constant $\delta$ to it. At the same time, the speaker decrements all the other in the first row and in the
first column by a smaller inhibiting constant $\epsilon$. It then follows the same procedure with the next active node, which is the fourth one. The process is illustrated in figure 8. The resulting utterance in this case is “AB”, and it is sent to the hearer.

The dual strengthening-inhibiting update rule is intended to discourage synonymy and polysemy. It is inspired by the mutual exclusivity bias seen in the language acquisition behavior of young children. This bias is demonstrated by the following kinds of experiments. A child is presented with three, say, objects, where the child knows the names of two of them. The experimenter then makes up a nonsense word and says something like, “Which one is the argy-bargy?” In this situation children will prefer to associate the novel word with the novel object, rather than assuming that the novel word is another name for one of the known objects (Markman & Wachtel, 1988). In the same spirit, when a speaker or hearer increments a particular location in its form-meaning matrix, it decrements all the other entries in that row and column by a small amount. This is a fairly standard algorithm from the literature for learning a form-meaning mapping (Oliphant, 1997; Steels & Kaplan, 2002; Kaplan, 2005).

The hearer, upon receiving an utterance, looks through the corresponding rows of its form-meaning matrix. The resulting utterance is AB.
meaning mapping to find the highest-valued column entries. The hearer uses these columns to construct a hidden-layer activation vector by setting the corresponding hidden layer node activations to 0.9, and setting the remaining hidden layer activations to 0.1. However, the hearer does not update its form-meaning matrix during this process. Those updates are delayed until the expected output for the current input vector is received. Then, using the backpropagation algorithm, the hearer constructs a second expected hidden-layer activation vector, which this time will be used for learning, not for interpretation. It treats this new vector as the expected meaning vector for the utterance it had received from the speaker. The hearer does not know which symbols in the utterance are to be associated with which hidden layer activations in the backpropagated hidden-layer activation vector. So it simply updates its form-meaning mapping using all possible (symbol, active hidden-layer node) pairs, using the same (δ, ε) rule as the speaker. Over time, encountering different utterances and meaning vectors will result in the right statistics for the form-meaning mapping.

Through this process the agents collectively discover which hypotheses solve the classification problem. We will show, below, that this process actually results in a highly efficient encoding. The population converges on a set of hypotheses which includes close to the minimum number required. This is a highly non-trivial learning problem, as they are learning not just a solution to the problem, but something close to the best solution to the problem.

5 Experiments

Now we show the results of a series of experiments that demonstrate the effects of collective learning on the emergent representations. In each case (except the first), we train a population of agents on the xor task, because the results are easy to visualize, and then we examine the learned representation. In each case, the neural networks have four hidden layer nodes, and the form-meaning mappings have four symbols to associate with the hidden layer nodes. In the figures below from the experiments, we show the four points and the learned straight lines (hypotheses) corresponding to the hidden layer nodes. As mentioned earlier, each hidden layer node can be thought of as representing a straight line (a hyperplane in two dimensions), and the weights on the connections to the node are the coefficients in the equation of that line: \( ax + by + c = 0 \). A line is assumed to label points on one side of it as positive, and points on the other side as negative. If we rotate any line through \( 180^\circ \), it will lie exactly on top of its original position, but the points previously labeled positive will now be negative, and vice versa.

We say that the symbol corresponding to a hidden layer node is expressed or uttered if the corresponding straight line is oriented in such a way that the given point is labeled as positive by this line. This happens when the hidden layer node is “active”, i.e. has an output greater than 0.5. This is how the utterances given in the tables in the following experiments are computed.
5.1 Driving simplicity: communication without task learning

In this experiment there is no classification task. The experiment is meant to show the pressure towards learnability in collective language emergence. At each step, the speaker is given a random two bit Boolean input vector, and the output of the speaker is treated as the expected output for the hearer. In other words, the hearer is trying to predict the speaker’s output. As the game proceeds, the agents converge upon a shared input-output mapping, and this mapping is always a “trivial” mapping. An example is shown in figure 9.

<table>
<thead>
<tr>
<th>Input</th>
<th>Label</th>
<th>Utterance</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0)</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>(0, 1)</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>(1, 0)</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>(1, 1)</td>
<td>0</td>
<td>C</td>
</tr>
</tbody>
</table>

Figure 9: The result of ungrounded language learning. The agents come to consensus on a trivially simple mapping, which assigns the same utterance, and same label, to all the points.

We see, in figure 9, that all the hyperplanes have been pushed off to one side. It so happens that hyperplanes A, B, and D are oriented so that they label all the points as 0, while hyperplane C has taken on the reverse orientation and labels all the points as 1. This is why the table shows C as the utterance for all the points. The decoder for all the agents, though, decodes this as the label 0 for each point, as shown in the table, and also by the empty (unfilled) circles in the figure.

Note that the utterance actually carries no information, as the symbol C is always expressed. In other simulation runs for this experiment utterances might include multiple symbols, but when this happens the same set of symbols is always expressed for all the points. Thus in every run there is no information transmitted, and the mappings are always trivial.

5.2 Driving complexity: task learning without communication

The second experiment shows the opposite phenomenon. It illustrates the pressure towards expressiveness in language emergence. In this experiment there is no communication between agents; they are each learning individually to solve the xor problem and inventing their own language to describe the result. Figure 10 shows an example of an individually learned mapping. The agents had four hidden layer nodes available to them, and this agent uses them all. While it solves the problem perfectly, as we can see the learned representation is overly complex since only two nodes are actually needed. Different agents learn different solutions, and each agent’s particular solu-
tion depends of course on the random initialization of the neural network weights. However, the minimal solution using only two hyperplanes is observed only by chance, and very rarely.

The agents in this experiment are not updating their form-meaning mappings and thus it does not really make sense to talk about their languages in this case. But, if we assume a uniform mapping that assigns symbol A to the first hidden layer node, B to the second and so on, we can derive what their language would have been if this solution had been adopted by all agents.

<table>
<thead>
<tr>
<th>Input</th>
<th>Label</th>
<th>Utterance</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0)</td>
<td>0</td>
<td>ABC</td>
</tr>
<tr>
<td>(0, 1)</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>(1, 0)</td>
<td>1</td>
<td>BD</td>
</tr>
<tr>
<td>(1, 1)</td>
<td>0</td>
<td>B</td>
</tr>
</tbody>
</table>

Figure 10: A learned solution to the xor problem, without communication between agents. Different agents learn different solutions, but they all generally learn overly complex solutions.

It ought to be pointed out that there isn’t really a “pressure” towards increasing complexity. There are many solutions to the given task that are more complex than the minimal solution, and a neural network is far more likely to find one of these overly complex solutions. However, it is important to try to find the minimal solution, because that is most likely to generalize well to new data points. In the xor problem, of course, this is not an issue since we are given all the points during training. However, for any realistic learning task, the agents have to learn from a restricted training set, and generalization is the key performance measure. In such a case it can be shown that simpler solutions generalize better (Blum & Langford, 2003). This is a version of Ockham’s razor, and often a complexity term is explicitly introduced into the objective function to try to force the learner to find a simpler solution that will generalize well. This technique is known as complexity regularization (Barron, 1991; Hochreiter & Schmidhuber, 1997; Lugosi & Zeger, 1996).

5.3 Finding balance: coupled task-communication learning

In the next experiment, we allow the agents to learn to communicate while also learning to solve the task. With this task-language coupling, the agents converge to a maximally simple form-meaning mapping that also solves the problem. In some (chance-determined) runs, agents develop languages with redundant symbols, and in some they do not; a redundancy example is shown in figure 11.

The population for this experiment consisted of only four agents. Figure 12 shows the learned form-meaning mappings of all the agents, as Hinton diagrams. A filled box indicates a positive value at that location and an empty box indicates a negative value. The size of a box is proportional
The Classification Game

<table>
<thead>
<tr>
<th>Input</th>
<th>Label</th>
<th>Utterance</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0)</td>
<td>0</td>
<td>ABCD</td>
</tr>
<tr>
<td>(0, 1)</td>
<td>1</td>
<td>ABD</td>
</tr>
<tr>
<td>(1, 0)</td>
<td>1</td>
<td>ABD</td>
</tr>
<tr>
<td>(1, 1)</td>
<td>0</td>
<td>BD</td>
</tr>
</tbody>
</table>

Figure 11: A learned solution to the xor problem, with communication between agents. All agents converge to the same solution. Even though they have four hidden layers nodes, they converge on a simpler solution that uses only two of the nodes.

to the magnitude of the value. There are several interesting things to note about these matrices. First, they all map symbols and hyperplanes uniquely to each other. Each row and column has a distinct maximum in each of the matrices. Second, they are all different (except the first and third). In other words, their private interpretation of symbols is different, even though they all understand each other and have perfect performance on the task. Thus while their task representations and public language are aligned, their private languages are different.

<table>
<thead>
<tr>
<th>MEANING FORM</th>
<th>MEANING FORM</th>
<th>MEANING FORM</th>
<th>MEANING FORM</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 0 1 2 3</td>
<td>A 0 1 2 3</td>
<td>A 0 1 2 3</td>
<td>A 0 1 2 3</td>
</tr>
</tbody>
</table>

Figure 12: The learned form-meaning matrices for each of the agents from experiment 4.

5.4 Coupled learning: the emergence of a holistic language

The language that was shown to emerge in the previous experiment is compositional. It makes use of multiple symbols, and combines them meaningfully to communicate about the labels of various points. One way to understand the emergence of this compositionality is that the agents assume a default label of negative, or 0, for the points. The symbol A means ‘positive’, and the symbol C means ‘negation’. Thus AC means ‘not positive’. The symbols B and D are redundant and not meaningful. Though this is an interesting and desirable outcome from the point of view of language
evolution, it is reasonable to ask whether this outcome is in some way “built in,” or whether it is truly emergent.

<table>
<thead>
<tr>
<th>Input</th>
<th>Label</th>
<th>Utterance</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0)</td>
<td>0</td>
<td>AC</td>
</tr>
<tr>
<td>(0, 1)</td>
<td>1</td>
<td>AB</td>
</tr>
<tr>
<td>(1, 0)</td>
<td>1</td>
<td>AD</td>
</tr>
<tr>
<td>(1, 1)</td>
<td>0</td>
<td>A</td>
</tr>
</tbody>
</table>

Figure 13: A learned solution to the \textit{xor} problem, with communication during learning. All agents converge to the same solution. They essentially memorize the points, assigning one hidden layer node to each point.

To show that it is, in fact, emergent, we present the following result. Figure 13 shows the outcome of a run with identical parameters as the previous experiment. However, this time we see the emergence of a holistic language. Each point has a unique symbol associated with it (one of the points has no symbol, as A is redundant). In effect, the agents have memorized the points. This is only possible in the case where the neural network is large enough in the sense of having enough hidden layer nodes to assign a unique node to each point. In any realistic problem, this is generally not the case. However, the role of cognitive capacity (represented here by the size of the hidden layer) in the emergence of compositionality and syntax has been studied theoretically; Nowak, Plotkin, and Jansen (2000) showed that when the number of words that agents must remember exceeds a threshold, the emergence of syntax is triggered. In our case, this threshold is defined by the number of hidden layer nodes. If the number of points that must be labeled exceeds the number of hidden layer nodes, the network must clearly resort to a compositional code to solve the task.
The Classification Game

Figure 15: Flat and sharp minima. Flat minima are more robust to perturbation.

Figure 14, again, shows the learned form-meaning mappings. Once more, we observe that the private languages of the agents are different, even though their task performance is perfect both in the role of speaker and of hearer.

6 Analysis

The intuition behind the phenomena observed in the experiments is that the fitness of a learned representation in collective learning is determined by two things: its accuracy at classifying the data, and its robustness or resistance to perturbation. These two forces tend to cause the agents to converge upon a flat minimum of the error surface. A flat minimum also corresponds to a low complexity encoding, because, intuitively, the weights of the neural network have to be specified with less precision (Hochreiter & Schmidhuber, 1997). All the weight vectors in the region corresponding to the flat minimum are equivalent with respect to error.

Figure 15 shows the difference between flat and sharp minima. The advantage of converging upon a flat minimum is that it offers better generalization. This can be justified using a minimum description length argument, as done by Hochreiter and Schmidhuber (1997), who explicitly constructed an objective function which is minimized at flat minima of the error function. Their algorithm looks for axis-aligned boxes (hypercuboids) in weight space, with maximal volume, that satisfy two “flatness conditions”. The first flatness condition is that the change in the outputs of the network due to a perturbation of the weights should be less than a given constant, $\epsilon$. The second flatness condition enforces equal flatness along all axial directions. From these conditions they derive the following rather formidable objective function (Hochreiter & Schmidhuber, 1997, eqn 3.1).
\[ E(w, D_0) = E(\text{net}(w), D_0) + \lambda B(w, X_0), \]  
where \[ B(w, X_0) = \sum_{x_p \in X_0} B(w, x_p), \]  
and \[ B(w, x_p) = \frac{1}{2}(-L \log e + \sum_{i,j} \log \sum_k \left( \frac{\partial o^k(w, x_p)}{\partial w_{ij}} \right)^2 \]  
\[ + L \log \left( \sum_k \left( \frac{\partial o^k(w, x_p)}{\partial w_{ij}} \right)^2 \right) \]  

Here \( E(\text{net}(w), D_0) \) is the squared error of the neural network with weights \( w \) on training data \( D_0 \). \( X_0 \) is just the input set (i.e., part of \( D_0 \)), and \( o^k(w, x_p) \) is the output of the \( k \)th output unit on input vector \( x_p \). \( L \) is the dimension of the weight-space.

This learning algorithm seems to work rather well, given that they make several approximations in its derivation. Details can be found in their paper. One particular limitation seems to be that they only look for axis-aligned boxes instead of general connected regions.

Our system is performing essentially the same computation, because we are also finding low-complexity encodings robust to perturbation. The robustness comes from the fact that when a hearer tries to use a speaker’s encoding rather than its own, the resulting weight update is more like a perturbation since it is not in the direction of the hearer’s own gradient. However our agents are not restricted to look for axis-aligned boxes. Further, the complexity terms are being computed implicitly in the population. The explicit objective function being minimized by our agents is simply the squared error, which makes it much simpler to implement as well.

The population of agents is initialized randomly, and so it starts out spread all over the weight space. The collective learning process leads to a contraction. As the agents try out each others’ internal representations (when they act as hearers), they move closer to each other in weight space.

When an agent plays the role of speaker, its weight updates tend to drive it towards the local minimum. However, when the agent plays the role of hearer, then the weight update represents a perturbation if the agent’s internal representation doesn’t match the speaker’s. This perturbation results in an increase in the error unless the agent is on a flat minimum. Since the agents are distributed uniformly through weight space, and they all interact with each other, each agent gets perturbed in many different directions. This causes them to find flat minima that are equally flat in many directions. Further, as they contract slowly, they tend to find flat regions that are large. Thus the population implicitly performs the computation that is being done explicitly by the Flat Minimum Search algorithm of Hochreiter and Schmidhuber. Further, nowhere do we have to put in a restriction that agents can only seek axis-aligned regions.
The Classification Game

(a) The training data.

(b) The testing data.

Figure 16: The training and testing data for the study of overfitting. The training set has 50 points and the testing set has 200. The data generation process is described in the text. The line through the middle is the true classifier. Thus, points with x-coordinate greater than zero are labeled positive, and those with x-coordinate less than zero are labeled negative. However, there is noise in the classification, and some points are mislabeled, as can be seen.

6.1 Solving the overfitting problem?

The discussion above suggests that our system should be able to avoid overfitting. We demonstrate this in the experiment below. This task is taken from (Hochreiter & Schmidhuber, 1997), who in turn took it from (Pearlmutter & Rosenfeld, 1991). The task is simple. Points are in two dimensions, and the output is positive if the x-coordinate of the point is positive, negative if not. Points are generated by sampling from a zero mean, unit standard deviation circular Gaussian. However, points are mislabeled with probability 0.05. Finally, to generate the actual inputs, another random variable, sampled from a zero mean, 0.15 standard deviation Gaussian is generated for each point and added to it. We generated fifty such points for the training set, and another two hundred points for the testing set. These are shown in figures 16(a) and 16(b). The line through the middle shows the true classifier. We can see that several points are mislabeled in each data set.

We trained a population of ten agents with and without communication, for five million time steps on this task. Each agent had five hidden layer nodes available to it, though clearly only one node is needed. When agents are not communicating with each other, we effectively have an average over ten independent runs. The learning curve for this case is shown in figure 17(a). This is a classic case of overfitting. Both the training and the testing learning curves drop initially. But after a point, the training learning curve continues to drop while the error on the testing set starts to
Figure 17: Learning curves without and with communication. Figure 17(a) is an example of overfitting, where the error initially drops, but eventually the testing error starts to increase while the training error continues to drop, as the learner starts to fit noise in the data. With communication, though, the learners are able to find the right solution even though it has higher training error, as shown in figure 17(b). This figure actually contains four curves, as it includes speaker and hearer learning curves on both the training and the testing data.

Overfitting is one of the long-standing problems in learning, and several approaches exist to try to counter it. Some are heuristic approaches, such as early stopping. Other approaches, such as weight decay, are derived by making specific assumptions.

We can calculate the entropy of an agent’s encoding of the input by counting the frequencies of all the symbols in the utterances it generates. If $p_i$ is the probability of seeing the $i^{th}$ symbol in an utterance by an agent, then the entropy or average code length in bits is given by,

$$H = - \sum_i p_i \log_2 p_i.$$  

We calculate this quantity for each agent and then average it across the population as a measure of the complexity of the solution found by the population. A comparison of the average code length with and without communication is shown in figure 18. We see that as we increase the number of hidden layer nodes available to the agents, the complexity of the solution found without
The Classification Game

![Figure 18](image)

Figure 18: Average code length increases essentially linearly with the number of hidden layer nodes when there is no communication. It stays more or less constant when agents learn to solve the task while also learning to communicate about it. Average code length was calculated as described in the text.

communication increases more or less linearly. However, when agents communicate with each other, the complexity stays essentially constant, and close to the optimal value (which is around 0.5 in this case).

One possible limitation is that if the agents get too close to each other in weight space, they might start behaving essentially like a single agent since they would no longer be able to perturb each other much, whereupon they might roll off the flat region into a sharp minimum, as standard backpropagation tends to do. Another possibility is that overfitting can occur if the agents fail to converge. If the error landscape is very rugged, for example, agents can get stuck in local minima and fail to find a shared flat minimum.

6.2 Gender discrimination in images of faces

Now we try to train a population on a more difficult classification task, that of discriminating gender from images of faces. We use the “med_students” database (Diaco, DiCarlo, & Santos, 2000), which contains four hundred images of faces, of which two hundred are male and two hundred female. Twenty images from each class were randomly chosen to make up the training set, and the remaining three hundred and sixty were put into the testing set. Figure 19 shows some of these images.

We reduced the images to size $32 \times 32$. The neural networks thus had 1024 inputs. We set each neural network to have 50 hidden layer nodes, and, to force them to extract “good” features, we made them reconstruct the image at the output of the neural network, in addition to classifying it.
Figure 19: Male and female faces from the med_students database. Images were reduced to $32 \times 32$. Twenty male and twenty female faces were randomly chosen to make up the training set. The remaining 360 images (180 from each class) were put into the testing set. Sample images from the training set and testing set are shown.

Thus they had 1025 outputs, of which the first one was treated as the classification.

Note that we are giving raw pixel data to the neural networks, without any preprocessing. This representation is much more difficult to learn from than, e.g., taking the first several principal components, because the raw values don’t correct for differences in background lighting, position of the face, etc. Even though the images are taken to minimize differences in orientation, size, etc., it is still significantly difficult to use the pixel data directly. In fact, when we try to use a standard neural network (i.e. the no communication case), it is completely unable to improve classification performance through learning: the error always remains around 50%.

When the agents use communication we get significant improvement, as shown by the learning curves in figure 20. The agents are unable to converge on a shared language, however (the hearer error remains high), at least in the time for which we ran the simulation. Further, there is some evidence of overfitting, as the speaker error on the testing set increases slightly after dropping initially. Despite these issues, we believe that it is impressive that communication enables the neural networks to significantly improve performance in a situation where individual neural networks fail to learn.
7 More complex languages

We have seen that the languages that emerged in the experiments so far have been holistic or compositional. They have no more complex structure, such as ordering constraints on the symbols. An utterance “BC” is interpreted in the same way as the utterance “CB”. Indeed, some agents may generate one while others generate the other, without any effect on performance. A very interesting and important question for language evolution is, what causes the emergence of (more complex) grammar?

Here we show that by the simple expedient of introducing time, we obtain emergent languages which show partial ordering of symbols. Thus, it seems that at least rudimentary syntax is possible without the need for any explicitly syntactic processing within the agents.

We use simple recurrent neural networks to learn a temporal version of the xor task. The temporal version is the same as the regular xor task, except that the two inputs are presented in sequence, in two time steps. The agent has to produce the label, and an utterance, after seeing both inputs. Each agent now has a simple recurrent network as its cognitive architecture instead of the feed-forward network used earlier. A simple recurrent network (Elman, 1990) consists of a two-layer neural network where the outputs of the hidden layer from the previous time step are appended to the new inputs. Thus the network is capable of maintaining some temporal state information. The presence of this loop in the architecture complicates training somewhat. We use one of the
most common algorithms for training simple recurrent networks, called *backpropagation through time* (Werbos, 1990). This involves unrolling the network in time, as shown in figure 21, so that it is effectively learning as a feed-forward network. When some expected outputs become available, the error is backpropagated through the unrolled network (backpropagating through time), and then the weights of the corresponding layers are averaged to “collapse” the network once more.

Figure 22 shows a language that emerges on the temporal xor task. The hidden layer hyperplanes are now in six-dimensional space (two inputs + four previous hidden layer activations), so we cannot easily draw them anymore. We see from the table, though, that the agents make a syntactic distinction between the two positive points by the ordering of the symbols. All agents produce the utterance “A B” for the point \((-1, 1)\), and the utterance “B A” for the point \((1, -1)\). This distinction is entirely a consequence of the structure of the task, even though both points belong to the same class. Note that the part of the utterance produced after a single time step still does not have ordering constraints. For example the utterance “AB CD” would be interpreted in the same way as the utterance “BA DC”, but not “CD AB”, where the space indicates a time step. However, the fact that a minimal ordering constraint emerges without the need to build in a syntactic processing module is a very interesting outcome. This possibility has been speculated upon many times (e.g., (Solé, 2005)), but never actually demonstrated, to our knowledge.

Simple recurrent neural networks have in the past been treated as equivalent to finite state au-
The Classification Game

<table>
<thead>
<tr>
<th>Input</th>
<th>Label</th>
<th>Utterance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time step 1</td>
<td>Time step 2</td>
<td>Time step 1</td>
</tr>
<tr>
<td>−1</td>
<td>−1</td>
<td>−1</td>
</tr>
<tr>
<td>−1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>−1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>−1</td>
</tr>
</tbody>
</table>

Figure 22: The language learned on the temporal xor task.

tomata (Omlin & Giles, 1996). However, more recently they have been shown to have an implicit bias to learn representations that correspond to context-free and even some context-sensitive grammars (Bodén & Wiles, 2001). Thus, an interesting question is, how complex can the task and language get before we have to introduce some syntactic processing in the agents? This presents an interesting direction for future work.

8 Discussion and future work

Grounded simulations of language evolution generally treat linguistic input as just another perceptual input. For example, in the mushroom-world simulations of Cangelosi et al. (Cangelosi, 1999; Cangelosi & Harnad, 2000; Cangelosi & Parisi, 1998), a population of neural networks was evolved on a foraging task. The task was to distinguish between four hypothetical types of mushrooms - edible, poisonous, both, and neither. The poisonous effects of a mushroom that was both edible and poisonous could be avoided by taking the appropriate action (“marking”). Thus, when an agent encountered a mushroom, it had to produce three outputs - eat/not eat, mark/not mark, return/not return, and also whether to vocalize these actions. The vocalizations were simply three other outputs of the neural network, which would be made available as inputs to any other agents in the vicinity. The agents also produced some outputs which were interpreted as movement commands. Inputs to the agents consisted of five environmental inputs, in addition to the three corresponding to vocalizations of a nearby agent. No distinction was made, at the input to the neural networks, between the environmental and linguistic inputs. Details of the environment aside, this is the prime distinction between their experiments and ours.

Our approach, besides having the technical justification spelled out in section 3, also has some philosophical inspiration behind it. Some philosophers believe that linguistic utterances correspond directly to meanings that are internal to the agent. Millikan (2004), for example, contends that words are primarily used for direct perception. Thus they are distinct from other perceptual inputs. Also, syntactic evaluations or evaluations of truth value occur “after the fact”. Language understanding flows from semantics to syntax, and not the other way around. We believe that our approach embodies this notion, and that our experiments, at least partially, bear out these ideas.
Semantics is a very complex area, with several interpretations. Our simple system has, perhaps, a relatively naive kind of semantics, where symbols correspond directly to hypotheses about the world. Yet, the fact that the correct hypotheses are discovered interactively makes the semantics interesting. This is not truly dynamic semantics, since that would involve context-sensitivity, but perhaps our experiments with recurrent neural networks point the way to incorporating that.

Our classification game is a part of what is now a tradition of “language games” to study artificial language evolution, pioneered by Luc Steels et al. in the series of Talking Heads experiments (Steels, Kaplan, McIntyre, & Looveren, 2002). One of Steels’ early experiments was actually somewhat similar to the ones described here (Steels, 1998). Steels studied a population of artificial agents playing a “discrimination game”. Two randomly chosen agents would take on the roles of speaker and hearer, and would be presented with a context consisting of a few simple shapes which they could sense using various feature detectors. The speaker would have to name one of the shapes, and the hearer would have to correctly identify (point to) that shape. As the game progressed, the agents developed binary classification trees to distinguish the objects from one another, and also developed a shared lexicon to communicate the names of the objects to each other. A name was assigned to a collection of features that was sufficient to distinguish an object in the contexts in which it had been observed. Steels did not examine the effect of collective learning on the emergent feature descriptions of objects. The agents did not, in any case, have the means to develop anything other than a holistic language, because there was no direct connection between the names and individual features.

There has been some theoretical work, though, on the emergence of structured language. (Nowak et al., 2000) have discussed the conditions which might trigger the emergence of syntax. They developed a population-dynamical model of word learning and showed that once the lexicon size exceeds a threshold, grammatical languages have higher fitness. They point out that many animals probably have a syntactic understanding of the world, but they haven’t developed syntactic communication systems in part because the number of things they need to communicate about is below the threshold that favors syntactic communication. This “need to communicate” is another aspect of the emergence of language that is not included in our model. Our experiment with recurrent neural networks could be interpreted as developing a syntactic understanding of the classification problem, because a recurrent network can be thought of as a grammar, as mentioned above. However, the agents attempt to communicate about every point that needs to be classified. Extending this work to include decision-making about whether to communicate is another interesting direction for future work.

Additionally, in our experiments we have seen that compositionality must appear when the task cannot be solved by using a single hypothesis and the neural network does not have enough nodes to memorize the data. Thus two kinds of holistic languages can be seen in our system. When a single hypothesis is sufficient to solve the problem, we have a language where a single symbol is used for a single class. This would be something like an animal giving an alarm call when any
The Classification Game

predator is detected. The second kind of holistic language we see in our experiments is described in experiment 5, where a single hypothesis is associated with each point. This corresponds to giving a unique name to each object in the domain of discourse. Thus our model has intrinsic reasons for the emergence of holistic and compositional languages, as opposed to the population level model of Nowak et al.

Kirby et al. have also given an account of the emergence of compositionality via their Iterated Learning Model (ILM) (Smith, Kirby, & Brighton, 2003; Kirby, 2007). The ILM models cultural transmission of language, for example from parents to children through successive generations. Kirby et al. show that since language must pass through the “bottleneck” of child language acquisition, the only languages that are stable are the ones that allow the construction of new valid utterances on the basis of known utterances. In other words, compositionality is favored by the need to learn quickly from a few samples. This is the closest model to ours, in phenomenological terms, and we discuss it in some more detail next.

8.1 The Bottleneck Question

We consider here a particular implementation of the Iterated Learning Model (ILM). Brighton (2002, 2005) created a population of learning agents that use Finite State Unification Transducers (FSUTs) to learn language. Each generation consists of a single agent who observes a random subset of the entire language of its parent. On the basis of this subset, the agent constructs a finite state machine capable of generating exactly the observed utterances.

This is not the end of learning, though. The agent has a bias for constructing an FSUT of minimum complexity. So it follows a procedure to minimize the number of states and transitions in the initially inferred FSUT, by combining states where possible. This minimized FSUT has greater expressive power than the initially inferred FSUT because it can infer the utterances for some meanings that it has not seen from its parent. Nonetheless, it still may not have complete coverage.

When this agent has to teach the language to its child, it may be faced with the need to express some meanings that are not expressible using its FSUT. In this case, the agent uses an invention procedure that tries to use the structure of its FSUT wherever possible, to produce a partial utterance. The utterance is “completed” randomly to express the given meaning.

Both the minimization of the FSUT and the structured invention procedure are necessary for the emergence of compositionality in this implementation of the ILM. Brighton (2005) points out that compositionality will not appear if either of these components are removed. The role of the bottleneck, thus, is that it provides the opportunity for an agent to add structure to the language. In each generation some of the randomness in the language is stripped away and replaced with structure through the invention procedure.

Griffiths and Kalish (2005) built an abstract Bayesian model of the ILM, in which agents are
Bayesian learners who have some prior distribution over possible languages, and compute a posterior from the data they obtain from their parents. An agent then samples its posterior to choose a language to communicate to its child. They showed that the stationary distribution of the resulting Markov process is identical to the prior distribution assumed by the agents. Thus, in a sense, the ILM is not creating a compositional language, beyond making the prior manifest.

This is not necessarily a problem for the ILM. The larger point made by the ILM, that structure in language can appear through cultural rather than biological evolution (and hence need not depend on an “innate” biological specification), is still valid. This is because in the ILM there is no requirement that the structure-driving learning bias must be language-specific (Kirby, Smith, & Brighton, 2004). In fact, since complexity minimization is such an important part of learning in general (to maximize predictive ability), there is good reason to believe that this learning bias is a very general feature of cognition.

It has been suggested that a better model would be for the agents to choose the language corresponding to the maximum of their posterior distributions, rather than just sampling from it (Kirby, Dowman, & Griffiths, 2007). However, it is only practical to compute the maximum in trivial cases where the distribution can be maintained explicitly. For any realistic language space, finding the maximum is highly complex. Hence, the agents have to resort to some estimation technique like gradient ascent, which will only get them to a local maximum. Further enhancements such as simulated annealing can lead to “high” maxima, but cannot guarantee finding the global maximum either. In fact, we can’t rule out the possibility that sometimes an agent may do very poorly simply because of an unfortunate random initialization. It seems, then, that the assumption made by Griffiths and Kalish that the agent samples from its posterior is a reasonable one, and their model does seem to be a good representation of Brighton’s ILM implementation.

Our model is similar to the ILM in the sense that the population of agents tends to converge upon a simple language, which, as we have discussed earlier, leads to better generalization. However it is not clear if the causal mechanisms that lead to this phenomenon are the same in both models. There is no clear component of the model that can be identified as a bottleneck. In fact, since the agents never see examples outside the training set, there is no direct pressure to generalize. The simplicity-functionality tradeoff emerges in spite of this. The agents also do not have an intrinsic MDL bias, as Brighton’s agents do.

One of the limitations of the ILM is that it idealizes a generation to consist of a single agent. It is well known, however, that children learn language to a large extent from their peers. Vogt (2005) has extended the ILM by combining it with the Talking Heads model, to include peer interaction. He suggests that this results in an implicit bottleneck effect. It is possible that there could be a similar effect operating in our model. We could also extend our model in an analogous manner, by considering generations of agents, that get different training sets for the same problem, or possibly even for different, but related, problems. In this way the classification game would solve the “bias problem” in the ILM, since this iterated classification game model could not be criticized for build-
ing in a preference for structured language. It might also offer a model of phenomena such as the invention of the Nicaraguan Sign Language, a grammatical sign language created by a community of deaf children in Nicaragua, based on the pidgin-like Lenguaje de Signos Nicaragüense that they were exposed to at home (Kegl & Iwata, 1989).

9 Conclusion

The main conclusion of this article is that collective learning involves a balance between learnability and functionality. It results in representations that are complex enough to solve the task at hand, yet simple enough to generalize well. We have introduced the classification game as a protocol, or collective learning algorithm, which allows a population of agents to interactively discover such a solution. We have shown that these agents can create a conventionalized public language by learning to associate a set of forms with the meanings they are collectively developing. The learning process generally results in compositional languages, but holistic languages can appear if the task is simple enough. By making the task temporal, we have also shown that rudimentary syntax, in the form of partial symbol ordering, can appear. This happens without the introduction of any specifically syntactic processing in the agents’ cognitive architecture. Rather, the use of a recurrent neural network essentially gives them a syntactic understanding of the problem, which is reflected in their language.

We have also presented an intuitive analysis of why collective learning results in complexity regularization, based on the notion of flat minima of the error surface. Formalizing this analysis is one of the main goals of future work. We have also suggested that collective learning might offer a novel solution to the overfitting problem, though possibly some kind of early stopping might still be necessary.

Finally, we have discussed some connections with other work in language evolution, and this has led to some interesting ideas for extending the current work.

10 Acknowledgements

We thank two anonymous reviewers, Paul Vogt, and the members of the Information Dynamics Lab at the University of Illinois at Urbana-Champaign for their comments, which helped to improve this paper.
References


The Classification Game

learning. In *Proceedings of the 27th annual conference of the cognitive science society*.


