Comparing learners for Boolean partitions: implications for morphological paradigms [*]

Katya Pertsova
University of North Carolina,
Chapel Hill
pertsova@email.unc.edu

Abstract

In this paper, I show that a problem of learning a morphological paradigm is similar to a problem of learning a partition of the space of Boolean functions. I describe several learners that solve this problem in different ways, and compare their basic properties.

1 Introduction

Lately, there has been a lot of work on acquiring paradigms as part of the word-segmentation problem (Zeman, 2007; Goldsmith, 2001; Snover et al., 2002). However, the problem of learning the distribution of affixes within paradigms as a function of their semantic (or syntactic) features is much less explored to my knowledge. This problem can be described as follows: suppose that the segmentation has already been established. Can we now predict what affixes should appear in what contexts, where by a 'context' I mean something quite general: some specification of semantic (and/or syntactic) features of the utterance. For example, one might say that the nominal suffix -z in English (as in *apple-z*) occurs in contexts that involve plural or possesive nouns whose stems end in a voiced segment.

In this paper, I show that the problem of learning the distribution of morphemes in contexts specified over some finite number of features is roughly equivalent to the problem of learning Boolean partitions of DNF formulas. Given this insight, one can easily extend standard DNF-learners to morphological paradigm learners. I show how this can be done on an example of the classical k-DNF learner (Valiant, 1984). This insight also allows us to bridge the paradigm-learning problem with other similar problems in the domain of cognitive science for which DNF's have been used, e.g., concept learning. I also describe two other learners proposed specifically for learning morphological paradigms. The first of these learners, proposed by me, was designed to capture certain empirical facts about syncretism and free variation in typological data (Pertsova, 2007). The second learner, proposed by David Adger, was designed as a possible explanation of another empirical fact - uneven frequencies of free variants in paradigms (Adger, 2006).

In the last section, I compare the learners on some simple examples and comment on their merits and the key differences among the algorithms. I also draw connections to other work, and discuss directions for further empirical tests of these proposals.

2 The problem

Consider a problem of learning the distribution of inflectional morphemes as a function of some set of features. Using featural representations, we can represent morpheme distributions in terms of a formula. The DNF formulas are commonly used for such algebraic representation. For instance, given the nominal suffix -z mentioned in the introduction, we can assign to it the following representation: $[(noun; +voiced]_{stem}; +plural) \vee (noun; +voiced]_{stem}; +possesive)]$. Presumably, features like $[plural]$ or $[+voiced]$ or $]_{stem}$ (end of the stem) are accessible to the learners' cognitive system, and can be exploited during the learning process for the purpose of "grounding" the distribution of morphemes.[1] This way of looking at things is similar to how some researchers conceive of concept-learning or word-

[*]This paper ows a great deal to the input from Ed Stabler. As usual, all the errors and shortcomings are entirely mine.

[1]Assuming an a priori given universal feature set, the problem of feature discovery is a subproblem of learning morpheme distributions. This is because learning what feature condition the distribution is the same as learning what features (from the universal set) are relevant and should be paid attention to.

learning (Siskind, 1996; Feldman, 2000; Nosofsky et al., 1994).

However, one prominent distinction that sets inflectional morphemes apart from words is that they occur in paradigms, semantic spaces defining a relatively small set of possible distinctions. In the absence of free variation, one can say that the affixes define a partition of this semantic space into disjoint blocks, in which each block is associated with a unique form. Consider for instance a present tense paradigm of the verb "to be" in standard English represented below as a partition of the set of environments over the following features: *class* (with values *masc, fem, both (masc & fem),inanim,*), $number$ (with values $+sg$ and $-sg$), and *person* (with values *1st, 2nd, 3rd*).[2]

am	1st. person; fem; +sg.
	1st. person; masc; +sg.
are	2nd. person; fem; +sg.
	2nd. person; masc; +sg.
	2nd. person; fem; −sg.
	2nd. person; masc; −sg.
	2nd. person; both; −sg.
	1st. person; fem; −sg.
	1st. person; masc; −sg.
	1st. person; both; −sg.
	3rd. person; masc; −sg
	3rd. person; fem; −sg
	3rd. person; both; −sg
	3rd. person; inanim; −sg
is	3rd person; masc; +sg
	3rd person; fem; +sg
	3rd person; inanim; +sg

Each block in the above partition can be represented as a mapping between the phonological form of the morpheme (a *morph*) and a DNF formula. A single morph will be typically mapped to a DNF containing a single conjunction of features (called a *monomial*). When a morph is mapped to a disjunction of monomials (as the morph [-z] discussed above), we think of such a morph as a homonym (having more than one "meaning"). Thus, one way of defining the learning problem is in terms of learning a partition of a set of DNF's.

[2]These particular features and their values are chosen just for illustration. There might be a much better way to represent the distinctions encoded by the pronouns. Also notice that the feature values are not fully independent: some combinations are logically ruled out (e.g. speakers and listeners are usually animate entities).

Alternatively, we could say that the learner has to learn a partition of Boolean functions associated with each morph (a Boolean function for a morph m maps the contexts in which m occurs to $true$, and all other contexts to $false$).

However, when paradigms contain free variation, the divisions created by the morphs no longer define a partition since a single context may be associated with more than one morph. (Free variation is attested in world's languages, although it is rather marginal (Kroch, 1994).) In case a paradigm contains free variation, it is still possible to represent it as a partition by doing the following:

(1) Take a singleton partition of morph-meaning pairs (m, r) and merge any cells that have the same meaning r. Then merge those blocks that are associated with the same set of morphs.

Below is an example of how we can use this trick to partition a paradigm with free-variation. The data comes from the past tense forms of "to be" in Buckie English.

was	1st. person; fem; +sg.
	1st. person; masc; +sg.
	3rd person; masc; +sg
	3rd person; fem; +sg
	3rd person; inanim; +sg
was/were	2nd. person; fem; +sg.
	2nd. person; masc; +sg.
	2nd. person; fem; −sg.
	2nd. person; masc; −sg.
	2nd. person; both; −sg.
	1st. person; fem; −sg.
	1st. person; masc; −sg.
	1st. person; both; −sg.
were	3rd. person; masc; −sg
	3rd. person; fem; −sg
	3rd. person; both; −sg
	3rd. person; inanim; −sg

In general, then, the problem of learning the distribution of morphs within a single inflectional paradigm is equivalent to learning a Boolean partition.

In what follows, I consider and compare several learners for learning Boolean partitions. Some of these learners are extensions of learners proposed in the literature for learning DNFs. Other learners were explicitly proposed for learning morphological paradigms.

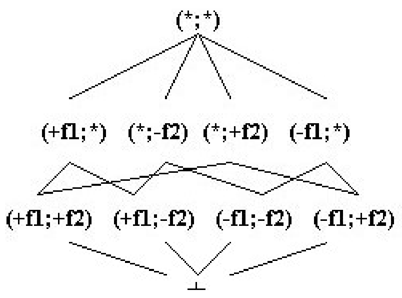
We should keep in mind that all these learners are idealizations and are not realistic if only because they are batch-learners. However, because they are relatively simple to state and to understand, they allow a deeper understanding of what properties of the data drive generalization.

2.1 Some definitions

Assume a finite set of morphs, Σ, and a finite set of features F. It would be convenient to think of morphs as chunks of phonological material corresponding to the pronounced morphemes.[3] Every feature $f \in F$ is associated with some set of values V_f that includes a value $[*]$, unspecified. Let S be the space of all possible complete assignments over F (an assignment is a set $\{f_i \rightarrow V_f | \forall f_i \in F\}$). We will call those assignments that do not include any unspecified features *environments*. Let the set $S' \subseteq S$ correspond to the set of environments.

It should be easy to see that the set S forms a Boolean lattice with the following relation among the assignments, $\leq_R$: for any two assignments a_1 and a_2, $a_1 \leq_R a_2$ iff the value of every feature f_i in a_1 is identical to the value of f_i in a_2, unless f_i is unspecified in a_2. The top element of the lattice is an assignment in which all features are unspecified, and the bottom is the contradiction. Every element of the lattice is a *monomial* corresponding to the conjunction of the specified feature values. An example lattice for two binary features is given in Figure 1.

Figure 1: A lattice for 2 binary features



A language L consists of pairs from $\Sigma \times S'$. That is, the learner is exposed to morphs in different environments.

[3]However, we could also conceive of morphs as functions specifying what transformations apply to the stem without much change to the formalism.

One way of stating the learning problem is to say that the learner has to learn a grammar for the target language L (we would then have to specify what this grammar should look like). Another way is to say that the learner has to learn the language mapping itself. We can do the latter by using Boolean functions to represent the mapping of each morph to a set of environments. Depending on how we state the learning problem, we might get different results. For instance, it's known that some subsets of DNF's are not learnable, while the Boolean functions corresponding to them are learnable (Valiant, 1984). Since I will use Boolean functions for some of the learners below, I introduce the following notation. Let B be the set of Boolean functions mapping elements of S' to *true* or *false*. For convenience, we say that b_m corresponds to a Boolean function that maps a set of environments to *true* when they are associated with m in L, and to *false* otherwise.

3 Learning Algorithms

3.1 Learner 1: an extension of the Valiant k-DNF learner

An observation that a morphological paradigm can be represented as a partition of environments in which each block corresponds to a mapping between a morph and a DNF, allows us to easily convert standard DNF learning algorithms that rely on positive and negative examples into paradigm-learning algorithms that rely on positive examples only. We can do that by iteratively applying any DNF learning algorithm treating instances of input pairs like (m, e) as positive examples for m and as negative examples for all other morphs.

Below, I show how this can be done by extending a k-DNF[4] learner of (Valiant, 1984) to a paradigm-learner. To handle cases of free variation we need to keep track of what morphs occur in exactly the same environments. We can do this by defining the partition Π on the input following the recipe in (1) (substituting environments for the variable r).

The original learner learns from negative examples alone. It initializes the hypothesis to the disjunction of all possible conjunctions of length at most k, and subtracts from this hypothesis monomials that are consistent with the negative examples. We will do the same thing for each

[4]k-DNF formula is a formula with at most k feature values in each conjunct.

morph using positive examples only (as described above), and forgoing subtraction in a cases of free-variation. The modified learner is given below. The following additional notation is used: Lex is the lexicon or a hypothesis. The formula D is a disjunction of all possible conjunctions of length at most k. We say that two assignments are *consistent* with each other if they agree on all specified features. Following standard notation, we assume that the learner is exposed to some text T that consists of an infinite sequence of (possibly) repeating elements from L. t_j is a finite subsequence of the first j elements from T. $L(t_j)$ is the set of elements in t_j.

```
Learner 1 (input: $t_j$)
    1. set $Lex := \{\langle m, D\rangle | \, \exists \langle m, e\rangle \in L(t_j)\}$
    2. For each $\langle m, e\rangle \in L(t_j)$, for each $m'$ s.t. $\neg\exists$ block $bl \in \Pi$ of $L(t_j)$,
       $\langle m, e\rangle \in bl$ and $\langle m', e\rangle \in bl$:
       replace $\langle m', f\rangle$ in $Lex$ by $\langle m', f'\rangle$ where $f'$ is the result of removing
       every monomial consistent with $e$.
```

This learner initially assumes that every morph can be used everywhere. Then, when it hears one morph in a given environment, it assumes that no other morph can be heard in exactly that environment unless it already knows that this environment permits free variation (this is established in the partition Π).

4 Learner 2:

The next learner is an elaboration on the previous learner. It differs from it in only one respect: instead of initializing lexical representations of every morph to be a disjunction of all possible monomials of length at most k, we initialize it to be the disjunction of all and only those monomials that are consistent with some environment paired with the morph in the language. This learner is similar to the DNF learners that do something on both positive and negative examples (see (Kushilevitz and Roth, 1996; Blum, 1992)).

So, for every morph m used in the language, we define a disjunction of monomials D_m that can be derived as follows. (i) Let E_m be the enumeration of all environments in which m occurs in L (ii) let M_i correspond to a set of all subsets of feature values in e_i, $e_i \in E$ (iii) let D_m be $\bigvee M$, where a set $s \in M$ iff $s \in M_i$, for some i.

Learner 2 can now be stated as a learner that is identical to Learner 1 except for the initial setting of Lex. Now, Lex will be set to $Lex := \{\langle m, D_m\rangle | \, \exists \langle m, e\rangle \in L(t_i)\}$.

Because this learner does not require enumeration of all possible monomials, but just those that are consistent with the positive data, it can handle "polynomially explainable" subclass of DNF's (for more on this see (Kushilevitz and Roth, 1996)).

5 Learner 3: a learner biased towards monomial and elsewhere distributions

Next, I present a batch version of a learner I proposed based on certain typological observations and linguists' insights about blocking. The typological observations come from a sample of verbal agreement paradigms (Pertsova, 2007) and personal pronoun paradigms (Cysouw, 2003) showing that majority of paradigms have either "monomial" or "elsewhere" distribution (defined below).

Roughly speaking, a morph has a monomial distribution if it can be described with a single monomial. A morph has an elsewhere distribution if this distribution can be viewed as a complement of distributions of other monomial or elsewhere-morphs. To define these terms more precisely I need to introduce some additional notation. Let $\bigcap e_x$ be the intersection of all environments in which morph x occurs (i.e., these are the invariant features of x). This set corresponds to a least upper bound of the environments associated with x in the lattice $\langle S, \leq_R\rangle$, call it lub_x. Then, let the *minimal monomial function* for a morph x, denoted mm_x, be a Boolean function that maps an environment to $true$ if it is consistent with lub_x and to $false$ otherwise. As usual, an extension of a Boolean function, $ext(b)$ is the set of all assignments that b maps to true.

(2) Monomial distribution
A morph x has a monomial distribution iff $b_x \equiv mm_x$.

The above definition states that a morph has a monomial distribution if its invariant features pick out just those environments that are associated with this morph in the language. More concretely, if a monomial morph always co-occurs with the feature +*singular*, it will appear in *all* singular environments in the language.

(3) Elsewhere distribution
A morph x has an elsewhere distribution iff $b_x \equiv mm_x - (mm_{x_1} \vee mm_{x_2} \vee \ldots \vee (mm_{x_n}))$ for all $x_i \neq x$ in Σ.

The definition above amounts to saying that a morph has an elsewhere distribution if the environments in which it occurs are in the extension of its minimal monomial function minus the minimal monomial functions of all other morphs. An example of a lexical item with an elsewhere distribution is the present tense form *are* of the verb "to be", shown below.

Table 1: The present tense of "to be" in English

	sg.	pl
1p.	am	are
2p.	are	are
3p.	is	are

Elsewhere morphemes are often described in linguistic accounts by appealing to the notion of blocking. For instance, the lexical representation of *are* is said to be unspecified for both person and number, and is said to be "blocked" by two other forms: *am* and *is*. My hypothesis is that the reason why such non-monotonic analyses appear so natural to linguists is the same reason for why monomial and elsewhere distributions are typologically common: namely, the learners (and, apparently, the analysts) are prone to generalize the distribution of morphs to minimal monomials first, and later correct any overgeneralizations that might arise by using default reasoning, i.e. by positing exceptions that override the general rule. Of course, the above strategy alone is not sufficient to capture distributions that are neither monomial, nor elsewhere (I call such distributions "overlapping", cf. the suffixes -*en* and -*t* in the German paradigm in Table 2), which might also explain why such paradigms are typologically rare.

Table 2: Present tense of some regular verbs in German

	sg.	pl
1p.	-e	-en
2p.	-st	-t
3p.	-t	-en

The original learner I proposed is an incremental learner that calculates grammars similar to those proposed by linguists, namely grammars consisting of a lexicon and a filtering "blocking" component. The version presented here is a simpler batch learner that learns a partition of Boolean functions instead.[5] Nevertheless, the main properties of the original learner are preserved: specifically, a bias towards monomial and elsewhere distributions.

To determine what kind of distribution a morph has, I define a relation *C*. A morph m stands in a relation *C* to another morph m' if $\exists\langle m, e\rangle \in L$, such that $lub_{m'}$ is consistent with e. In other words, mCm' if m occurs in any environment consistent with the invariant features of m'. Let C^+ be a transitive closure of C.

Learner 3 (input: t_j)

1. Let $S(t_j)$ be the set of pairs in t_j containing monomial- or elsewhere-distribution morphs. That is, $\langle m, e\rangle \in S(t_j)$ iff $\neg\exists m'$ such that mC^+m' and $m'C^+m$.
2. Let $O(t_j) = t_j - S(t_j)$ (the set of all other pairs).
3. A pair $\langle m, e\rangle \in S$ is a *least element* of S iff $\neg\exists\langle m', e'\rangle \in (S - \{\langle m, e\rangle\})$ such that $m'C^+m$.
4. Given a hypothesis Lex, and for any expression $\langle m, e\rangle \in Lex$: let $rem((m, e), Lex) = (m, (mm_m - \{b|\langle m', b\rangle \in Lex\}))$[6]

```
1. set $S := S(t_j)$ and $Lex := \emptyset$
2. While $S \neq \emptyset$: remove a *least* $x$ from $S$ and set $Lex := Lex \cup rem(x, Lex)$
3. Set $Lex := Lex \cup O(t_j)$.
```

This learner initially assumes that the lexicon is empty. Then it proceeds adding Boolean functions corresponding to minimal monomials for morphs that are in the set $S(t_j)$ (i.e., morphs that have either monomial or elsewhere distributions). This

[5]I thank Ed Stabler for relating this batch learner to me (p.c.).

[6]For any two Boolean functions b, b': $b - b'$ is the function that maps e to 1 iff $e \in ext(b)$ and $e \notin ext(b')$. Similarly, $b + b'$ is the function that maps e to 1 iff $e \in ext(b)$ and $e \in ext(b')$.

is done in a particular order, namely in the order in which the morphs can be said to block each other. The remaining text is learned by rote-memorization. Although this learner is more complex than the previous two learners, it generalizes fast when applied to paradigms with monomial and elsewhere distributions.

5.1 Learner 4: a learner biased towards shorter formulas

Next, I discuss a learner for morphological paradigms, proposed by another linguist, David Adger. Adger describes his learner informally showing how it would work on a few examples. Below, I formalize his proposal in terms of learning Boolean partitions. The general strategy of this learner is to consider simplest monomials first (those with the fewer number of specified features) and see how much data they can unambiguously and non-redundantly account for. If a monomial is consistent with several morphs in the text - it is discarded unless the morphs in question are in free variation. This simple strategy is reiterated for the next set of most simple monomials, etc.

```
Learner 4 (input $t_j$)

1. Let $M_i$ be the set of all monomials over $F$ with $i$ specified features.

2. Let $B_i$ be the set of Boolean functions from environments to truth values corresponding
   to $M_i$ in the following way: for each monomial $mn \in M_i$ the corresponding Boolean
   function $b$ is such that $b(e) = 1$ if $e$ is an environment consistent with $mn$;
   otherwise $b(e) = 0$.

3. Uniqueness check:
   For a Boolean function $b$, morph $m$, and text $t_j$ let $unique(b, m, t_j) = 1$ iff
   $ext(b_m) \subseteq ext(b)$ and $\neg\exists\langle m', e\rangle \in L(t_j)$, s.t.
   $e \in ext(b)$ and $e \notin ext(b_m)$.

   1. set $Lex := \Sigma \times \emptyset$ and $i := 0$;

   2. while $Lex$ does not correspond to $L(t_j)$ AND $i \leq |F|$ do:
      for each $b \in B_i$, for each $m$, s.t. $\exists\langle m, e\rangle \in L(t_j)$:

         • if $unique(b, m, t_j) = 1$ then
           replace $\langle m, f\rangle$ with $\langle m, f + b\rangle$ in $Lex$

      $i \leftarrow i + 1$
```

This learner considers all monomials in the order of their simplicity (determined by the number of specified features), and if the monomial in question is consistent with environments associated with a unique morph then these environments are added to the extension of the Boolean function for that morph. As a result, this learner will converge faster on paradigms in which morphs can be described with disjunctions of shorter monomials since such monomials are considered first.

6 Comparison

6.1 Basic properties

First, consider some of the basic properties of the learners presented here. For this purpose, we will assume that we can apply these learners in an iterative fashion to larger and larger batches of data. We say that a learner is *consistent* if and only if, given a text t_j, it always converges on the grammar generating all the data seen in t_j (Osherson et al., 1986). A learner is *monotonic* if and only if for every text t and every point $j < k$, the hypothesis the learner converges on at t_j is a subset of the hypothesis at t_k (or for learners that learn by elimination: the hypothesis at t_j is a superset of the hypothesis at t_k). And, finally, a learner is *generalizing* if and only if for some t_j it converges on a hypothesis that makes a prediction beyond the elements of t_j.

The table below classifies the four learners according to the above properties.

Learner	consist.	monoton.	generalizing
Learner 1	yes	yes	yes
Learner 2	yes	yes	yes
Learner 3	yes	no	yes
Learner 4	yes	yes	yes

All learners considered here are generalizing and consistent, but they differ with respect to monotonicity. Learner 3 is non-monotonic while the remaining learners are monotonic. While monotonicity is a nice computational property, some aspects of human language acquisition are suggestive of a non-monotonic learning strategy, e.g. the presence of overgeneralization errors and their subsequent corrections by children(Marcus et al., 1992). Thus, the fact that Learner 3 is non-monotonic might speak in its favor.

6.2 Illustration

To demonstrate how the learners work, consider this simple example. Suppose we are learning the following distribution of morphs A and B over 2 binary features.

(4) Example 1

	+f1	−f1
+f2	A	B
−f2	B	B

Suppose further that the text t_3 is:

A	$+f1; +f2$
B	$-f1; +f2$
B	$+f1; -f2$

Learner 1 generalizes right away by assuming that every morph can appear in every environment which leads to massive overgeneralizations. These overgeneralizations are eventually eliminated as more data is discovered. For instance, after processing the first pair in the text above, the learner "learns" that B does not occur in any environment consistent with $(+f1; +f2)$ since it has just seen A in that environment. After processing t_3, Learner 1 has the following hypothesis:

A	$(+f1; +f2) \vee (-f1; -f2)$
B	$(-f1) \vee (-f2)$

That is, after seeing t_3, Learner 2 correctly predicts the distribution of morphs in environments that it has seen, but it still predicts that both A and B should occur in the not-yet-observed environment, $(-f1; -f2)$. This learner can sometimes converge before seeing all data-points, especially if the input includes a lot of free variation. If fact, if in the above example A and B were in free variation in all environments, Learner 1 would have converged right away on its initial setting of the lexicon. However, in paradigms with no free variation convergence is typically slow since the learner follows a very conservative strategy of learning by elimination.

Unlike Learner 1, Learner 2 will converge after seeing t_3. This is because this learner's initial hypothesis is more restricted. Namely, the initial hypothesis for A includes disjunction of only those monomials that are consistent with $(+f1; +f2)$. Hence, A is never overgeneralized to $(-f1; -f2)$. Like Learner 1, Learner 2 also learns by elimination, however, on top of that it also restricts its initial hypothesis which leads to faster convergence.

Let's now consider the behavior of learner 3 on example 1. Recall that this learner first computes minimal monomials of all morphs, and checks in they have monomial or elsewhere distributions (this is done via the relation C^+). In this case, A has a monomial distribution, and B has an elsewhere distribution. Therefore, the learner first computes the Boolean function for A whose extension is simply $(+f1; +f2)$; and then the Boolean function for B, whose extension includes environments consistent with (*;*) minus those consistent with $(+f1; +f2)$, which yields the following hypothesis:

$ext(b_A)$	$[+f1; +f2]$
$ext(b_B)$	$[-f1; +f2][+f1; -f2][-f1; -f2]$

That is, Learner 3 generalizes and converges on the right language after seeing text t_3.

Learner 4 also converges at this point. This learner first considers how much data can be unambiguously accounted for with the most minimal monomial (*;*). Since both A and B occur in environments consistent with this monomial, nothing is added to the lexicon. On the next round, it considers all monomials with one specified feature. 2 such monomials, $(-f1)$ and $(-f2)$, are consistent only with B, and so we predict B to appear in the not-yet-seen environment $(-f1; -f2)$. Thus, the hypothesis that Learner 4 arrives at is the same as the hypothesis Learners 3 arrives at after seeing t_3.

6.3 Differences

While the last three learners perform similarly on the simple example above, there are significant differences between them. These differences become apparent when we consider larger paradigms with homonymy and free variation.

First, let's look at an example that involves a more elaborate homonymy than example 1. Consider, for instance, the following text.

(5) Example 2

A	$[+f1; +f2; +f3]$
A	$[+f1; -f2; -f3]$
A	$[+f1; +f2; -f3]$
A	$[-f1; +f2; +f3]$
B	$[-f1; -f2; -f3]$

Given this text, all three learners will differ in their predictions with respect to the environment $(-f1;+f2;-f3)$. Learner 2 will predict both A and B to occur in this environment since not enough monomials will be removed from representations of A or B to rule out either morph from occurring in $(-f1;+f2;-f3)$. Learner 3 will predict A to appear in all environments that haven't been seen yet, including $(-f1;+f2;-f3)$. This is because in the current text the minimal monomial for A is $(*;*;*)$ and A has an elsewhere distribution. On the other hand, Learner 4 predicts B to occur in $(-f1;+f2;-f3)$. This is because the extension of the Boolean function for B includes any environments consistent with $(-f1;-f3)$ or $(-f1;-f2)$ since these are the simplest monomials that uniquely pick out B.

Thus, the three learners follow very different generalization routes. Overall, Learner 2 is more cautious and slower to generalize. It predicts free variation in all environments for which not enough data has been seen to converge on a single morph. Learner 3 is unique in preferring monomial and elsewhere distributions. For instance, in the above example it treats A as a 'default' morph. Learner 4 is unique in its preference for morphs describable with disjunction of simpler monomials. Because of this preference, it will sometimes generalize even after seeing just one instance of a morph (since several simple monomials can be consistent with this instance alone).

One way to test what the human learners do in a situation like the one above is to use artificial grammar learning experiments. Such experiments have been used for learning individual concepts over features like shape, color, texture, etc. Some work on concept learning suggests that it is subjectively easier to learn concepts describable with shorter formulas (Feldman, 2000; Feldman, 2004). Other recent work challenges this idea (Lafond et al., 2007), showing that people don't always converge on the most minimal representation, but instead go for the more simple and general representation and learn exceptions to it (this approach is more in line with Learner 3).

Some initial results from my pilot experiments on learning partitions of concept spaces (using abstract shapes, rather than language stimuli) also suggest that people find paradigms with elsewhere distributions easier to learn than the ones with overlapping distributions (like the German paradigms in 2). However, I also found a bias toward paradigms with the fewer number of relevant features. This bias is consistent with Learner 4 since this learner tries to assume the smallest number of relevant features possible. Thus, both learners have their merits.

Another area in which the considered learners make somewhat different predictions has to do with free variation. While I can't discuss this at length due to space constraints, let me comment that any batch learner can easily detect free-variation before generalizing, which is exactly what most of the above learners do (except Learner 3, but it can also be changed to do the same thing). However, since free variation is rather marginal in morphological paradigms, it is possible that it would be rather problematic. In fact, free variation is more problematic if we switch from the batch learners to incremental learners.

7 Directions for further research

There are of course many other learners one could consider for learning paradigms, including approaches quite different in spirit from the ones considered here. In particular, some recently popular approaches conceive of learning as matching probabilities of the observed data (e.g., Bayesian learning). Comparing such approaches with the algorithmic ones is difficult since the criteria for success are defined so differently, but it would still be interesting to see whether the kinds of prior assumptions needed for a Bayesian model to match human performance would have something in common with properties that the learners considered here relied on. These properties include the disjoint nature of paradigm cells, the prevalence of monomial and elsewhere morphs, and the economy considerations. Other empirical work that might help to differentiate Boolean partition learners (besides typological and experimental work already mentioned) includes finding relevant language acquisition data, and examining (or modeling) language change (assuming that learning biases influence language change).

References

David Adger. 2006. Combinatorial variation. *Journal of Linguistics*, 42:503–530.

Avrim Blum. 1992. Learning Boolean functions in an infinite attribute space. *Machine Learning*, 9:373–386.

Michael Cysouw. 2003. *The Paradigmatic Structure of Person Marking*. Oxford University Press, NY.

Jacob Feldman. 2000. Minimization of complexity in human concept learning. *Nature*, 407:630–633.

Jacob Feldman. 2004. How surprising is a simple pattern? Quantifying 'Eureka!'. *Cognition*, 93:199–224.

John Goldsmith. 2001. Unsupervised learning of a morphology of a natural language. *Computational Linguistics*, 27:153–198.

Anthony Kroch. 1994. Morphosyntactic variation. In Katharine Beals et al., editor, *Papers from the 30th regional meeting of the Chicago Linguistics Society: Parasession on variation and linguistic theory*. Chicago Linguistics Society, Chicago.

Eyal Kushilevitz and Dan Roth. 1996. On learning visual concepts and DNF formulae. *Machine Learning*, 24:65–85.

Daniel Lafond, Yves Lacouture, and Guy Mineau. 2007. Complexity minimization in rule-based category learning: revising the catalog of boolean concepts and evidence for non-minimal rules. *Journal of Mathematical Psychology*, 51:57–74.

Gary Marcus, Steven Pinker, Michael Ullman, Michelle Hollander, T. John Rosen, and Fei Xu. 1992. Overregularization in language acquisition. *Monographs of the Society for Research in Child Development*, 57(4). Includes commentary by Harold Clahsen.

Robert M. Nosofsky, Thomas J. Palmeri, and S.C. McKinley. 1994. Rule-plus-exception model of classification learning. *Psychological Review*, 101:53–79.

Daniel Osherson, Scott Weinstein, and Michael Stob. 1986. *Systems that Learn*. MIT Press, Cambridge, Massachusetts.

Katya Pertsova. 2007. *Learning Form-Meaning Mappings in the Presence of Homonymy*. Ph.D. thesis, University of California, Los Angeles.

Jeffrey Mark Siskind. 1996. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, 61(1-2):1–38, Oct-Nov.

Matthew G. Snover, Gaja E. Jarosz, and Michael R. Brent. 2002. Unsupervised learning of morphology using a novel directed search algorithm: taking the first step. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning*, pages 11–20, Morristown, NJ, USA. Association for Computational Linguistics.

Leslie G. Valiant. 1984. A theory of the learnable. *CACM*, 17(11):1134–1142.

Daniel Zeman. 2007. Unsupervised acquiring of morphological paradigms from tokenized text. In *Working Notes for the Cross Language Evaluation Forum*, Budapest. Madarsko. Workshop.