Learning Local Transductions is Hard

Martin Jansche (`jansche@acm.org`)
Center for Computational Learning Systems, Columbia University

Abstract. Local deterministic string-to-string transductions arise in natural language processing tasks such as letter-to-sound translation or pronunciation modeling. This class of transductions is a simple generalization of morphisms of free monoids; learning local transductions is essentially the same as inference of certain monoid morphisms. However, learning even a highly restricted class of morphisms, the so-called fine morphisms, leads to intractable problems: deciding whether a hypothesized fine morphism is consistent with observations is an NP-complete problem; and maximizing classification accuracy of the even smaller class of alphabetic substitution morphisms is APX-hard. These theoretical results provide some justification for using the kinds of heuristics that are commonly used for this learning task.

Keywords: Machine learning, formal languages, rational transductions, natural language processing, letter-to-sound rules, NP completeness, combinatorial optimization, Boolean satisfiability

1. Introduction

The use of machine learning has impacted the software engineering of many natural language processing (NLP) applications considerably. The involvement of domain experts has shifted away from explicit knowledge representation and toward the creation of labeled data sets that machine learning algorithms then generalize over in an attempt to distill the implicitly given expert knowledge. The machine learning approach is generally seen as desirable for many kinds of applications, since it removes the need on the part of domain experts to reason about the various declarative and procedural aspects of the given knowledge representation formalism. Different ways of representing expert knowledge can be explored straightforwardly by employing different kinds of learning algorithms, and the choice of a learning algorithm with an efficient representation of the inferred concepts can lead to considerable performance gains (Jansche, 2001).

This paper is concerned with the formal underpinnings of certain NLP applications built on very simple translation tasks that can be adequately modeled by finite state transducers (Mohri, 1997; Roche and Schabes, 1997). Approaches to NLP based on finite automata suffer from a shortage of theoretically sound and practically feasible learning algorithms for inferring automata from data. This is partly due to the fact that many classes of finite automata cannot be learned (efficiently) from positive data under most conceptualizations of learning (Gold, 1967; Pitt, 1989).

Some leverage can be gained by working with proper subclasses of regular languages or transductions (Angluin, 1982; García and Vidal, 1990; Oncina et al., 1993). We focus in particular on deterministic transducers which are local in nature, i. e., which consider some fixed, finite input context when deciding what output to generate. Local transducers, as defined below, are a highly restricted subclass of the generalized sequential machines (Eilenberg, 1974) and form the basis of a number of NLP tasks, including spelling correction, letter-to-sound rules (see, for example, van den Bosch, 1997), pronunciation modeling (see, for example, Gildea and Jurafsky, 1996), etc.

We will formulate the underlying learning problems associated with identifying local transducers in terms of abstract optimization problems and provide an analysis of their complexity. The key result presented here is that efficient inference of local transducers is, under certain assumptions, impossible, even in highly restricted cases. We show that deciding whether there are any local transducers consistent with a given set of training samples is an NP-complete problem. This result affects all learning approaches that rely on finding consistent hypotheses or on minimizing training error.

We proceed as follows: Section 2 provides further details on the general learning problem and its applications and carves out the core problems a learning algorithm needs to solve. Section 3 analyzes the computational complexity of the core problems and discusses the implications for the learnability of local transductions. Section 4 concludes and points out directions for further research.

2. The learning problem

2.1. Practical application: Letter-to-sound rules

The learning problem we are concerned with is illustrated by a specific NLP task, namely learning letter-to-sound rules (see, for example, Lucassen and Mercer, 1984; Sejnowski and Rosenberg, 1987; van den Bosch, 1997) by generalizing over a pronunciation dictionary, under certain locality assumptions discussed below. In this task the training samples are pairs of strings, consisting of a string of letters, for example ⟨shoes⟩, and a string of phonemes, for example /ʃuz/. Note that no positional correspondence or other relation between individual letters and phonemes is specified, which is to say one does generally *not* know whether e. g. the second symbol in /ʃuz/, the phoneme /u/, corresponds to the second symbol in ⟨shoes⟩, the letter ⟨h⟩. In this sense the present task is markedly different from some other common NLP tasks, such as part-of-speech assignment, where explicit correspondences between input and output symbols exist.

In general, a letter string may correspond to a longer phoneme string, for example,[1]

⟨mutualism⟩ (9 letters)
/mjuʧəwəlɪzəm/ (12 phonemes),

or to a shorter phoneme string, such as

⟨featherweight⟩ (13 letters)
/fɛðɚwet/ (7 phonemes);

and even if the two strings happen to have the same length, as in

⟨parliamentarianism⟩ (18 letters)
/paɹləməntɛɹiənɪzəm/ (18 phonemes),

no meaningful positional correspondences are implied. Most existing approaches assume that letter strings are of equal length or longer than their corresponding phoneme strings. While clearly false in an absolute sense, this assumption is true for most English words – it holds for more than 98% of the entries in the CMU pronouncing dictionary – and workarounds for cases where it seems to break down have been suggested, for example the transcription conventions used by NETtalk (Sejnowski and Rosenberg, 1987). We will adopt the same simplifying assumption here, since it allows us to treat the mapping from letter strings to phoneme strings as involving only deletions and substitutions, but not insertions.

Learning letter-to-sound rules can be conceptualized as grammatical inference of specific subclasses of rational transductions. For the class of subsequential transductions, limit-identification is possible, as Oncina et al. (1993) have shown. Their learning algorithm has been applied to the closely related problem of phonemic modeling (Gildea and Jurafsky, 1996), but only after modifications and incorporation of domain-specific knowledge. It can be shown (Jansche, 2003) that the algorithm proposed by Oncina et al. (1993) has poor out-of-class behavior and is brittle in the presence of imperfect data; furthermore its hypothesis space, the class of subsequential transductions, is arguably too general for the present task. Almost all approaches to learning letter-to-sound rules assume, justifiably (Lucassen and Mercer, 1984), that the hypothesis space is restricted to the analog of the locally testable languages in the strict sense (McNaughton and Papert, 1972), which are limit-identifiable (García and Vidal, 1990). We call the analogous class of transductions *local transductions*. Local transductions are a proper subclass of the subsequential transductions, and as such are limit-identifiable (Oncina et al., 1993).

Just as locally testable languages are accepted by scanner automata, local transductions are computed by scanner transducers, which move a sliding window of fixed size across the input string and produce a string of output

[1] The following examples are taken from the CMU pronouncing dictionary (Weide, 1998). The original phonemic transcription system has been changed to IPA (International Phonetic Association, 1999).

symbols for each window position; concatenating these output strings yields the overall output of the transducer. For example, the letter string ⟨shoes⟩ might be processed by a scanner with a three-letter window, which encounters the following substrings and produces an output phoneme for each window position:

#sh	sho	hoe	oes	es#
ʃ			u	z

Since the size of the sliding window is fixed and known a priori, one may assume without loss of generality that it is 1: if a larger window size n is needed, one can simply change the input alphabet to consist of n-tuples of symbols, and such a modified alphabet is obviously still finite. Alternatively, one can think of this modification as a preprocessing step that applies a simple subsequential transducer to each input string. In our running example, this would mean that the string of letters

⟨s h o e s⟩

is first deterministically transformed by a subsequential transducer into the string of letter triples

⟨#sh sho hoe oes es#⟩,

which is then handed to a scanner with a sliding window of size 1. Note that this preprocessing step is an isomorphic mapping which preserves string length. A restricted scanner with a one-symbol window can then do the same work done by the three-letter scanner earlier in this example. In other words, the accumulation of input context can be treated as a deterministic preprocessing step and does not affect the core of the learning problem. The learning problem itself has been simplified, since we only need to concern ourselves with transductions that operate on input strings symbol by symbol.

2.2. Morphisms of free monoids

Sequential transductions that examine individual input symbols (individual letters, or n-tuples of letters after preprocessing) without taking any context into account can be realized by generalized sequential machines with a trivial one-state topology and correspond exactly to morphisms of free monoids (Eilenberg, 1974, p. 299). The subsequent discussion will refer to a finite set Σ of input symbols and a finite set Γ of output symbols. The free monoid generated by Σ is called $\langle \Sigma^*, \cdot \rangle$, or Σ^* for short, and has the property that every element (string) $x \in \Sigma^*$ has a unique factorization in terms of elements (symbols) of Σ. This means that a morphism $g : \Sigma^* \to \Gamma^*$ from the free monoid Σ^* to the free monoid Γ^* is completely characterized by $g|_\Sigma$, the restriction of g to the input alphabet Σ. Conversely, this allows us to define the following notion:

DEFINITION 1 (Free monoid morphism). Given a function $f : \Sigma \to \Gamma^*$ define f^* to be the unique monoid morphism $f^* : \Sigma^* \to \Gamma^*$ such that $f^*(x) = f(x)$ for all $x \in \Sigma$; moreover f^* preserves the monoid's product $f^*(yz) = f^*(y)\ f^*(z)$ (for all $y, z \in \Sigma^*$) and unit element $f^*(\varepsilon) = \varepsilon$.

At the core of the learning task is then the problem of finding a suitable function $f : \Sigma \to \Gamma^*$ mapping from individual input symbols to output strings. In this paper we focus on two classes of functions. The first class restricts the codomain to strings of length one. If $f : \Sigma \to \Gamma$ is such a function – an alphabetic substitution – then f^* is a *very fine morphism*, according to Eilenberg (1974: 6). The second class is a superset of the first and allows the empty string in the codomain. Eilenberg (1974) calls the morphism f^* a *fine morphism* if its underlying function f is of type $\Sigma \to \{\varepsilon\} \cup \Gamma$. For the specific problem of learning letter to sound rules, we restrict our attention to fine morphisms, since by our previous assumption letter strings are never shorter than their corresponding phoneme strings, so a fine morphism is formally adequate. In general we may want to consider other kinds of morphisms, for example those arising from functions of type $\Sigma \to \{\varepsilon\} \cup \Gamma \cup \Gamma^2$. However, most practically relevant classes of morphisms will contain the class of fine morphisms, and therefore the problems arising from the use of fine morphisms will carry over to more general settings. By restricting our attention to fine morphisms we have narrowed down the initial learning task considerably, as the hypothesis space H is now the set of functions of type $\Sigma \to \{\varepsilon\} \cup \Gamma$, which is always finite (though usually very large) for fixed finite alphabets Σ and Γ. Moreover, since $\log |H| = |\Sigma| \log(1 + |\Gamma|)$ the sample complexity for this hypothesis space is polynomial.

2.3. Consistent hypotheses and loss functions

Polynomial sample complexity is one of the two key ingredients for PAC learnability (Valiant, 1984; Kearns and Vazirani, 1994), the other being computational complexity, which is the topic of the next section. However, the present problem does not meet all conditions of the PAC learning framework, which assumes the existence of consistent hypotheses. If a consistent hypothesis could be found for a sufficiently large training set – in other words, if there were a fine morphism consistent with all entries in a sufficiently large training dictionary – the PAC learning framework would give us certain guarantees about the quality of such a hypothesis.

The assumption of consistent hypothesis is not exclusive to PAC learning. In fact, most classical conceptualizations of learning are based on the same assumption, including identification in the limit (Gold, 1967). That is to say, these frameworks assume that a learner which can identify a given concept class will be presented with clean training data which illustrate a concept from that same class.

Unfortunately, this assumption is untenable in many practical situations, since the details of the process which generated the training data are generally unknown. In the case of letter-to-sound rules, for example, Lucassen and Mercer (1984) have demonstrated empirically that the transduction process is overwhelmingly local. However, there are no theoretical guarantees of locality, so modeling letter-to-sound rules with local transductions has to be viewed as an approximation, albeit a very good one. A learner should be able to gracefully deal with deviations from this assumption, regardless of whether they arise from real deficiencies of the model (i. e., the real transduction process is actually not local) or from deficiencies of the training data due to inconsistencies.

In other words, in practice one is often interested in finding the 'best' hypothesis, regardless of whether it is consistent with the training data. In this situation learning is better conceptualized as empirical risk minimization (but see Minka, 2000), or more formally as agnostic learning (Kearns et al., 1992).

Here we focus exclusively on empirical risk minimization, which says that the 'best' hypothesis is one whose empirical risk is minimal among all competing hypotheses. In our case, the empirical risk $R(h)$ of a hypothesis $h : \Sigma^* \to \Gamma^*$ is its average loss on a (multi)set of training samples $D \subseteq \Sigma^* \times \Gamma^*$, namely

$$R(h) = \frac{1}{|D|} \sum_{\langle x,y \rangle \in D} L(h(x), y)$$

where $L : \Gamma^* \times \Gamma^* \to \mathbb{Q}_{\geq 0}$ is a so-called *loss function*, assigning a non-negative rational number (loss) to each pair of output strings. The most commonly used generally applicable loss functions for comparing strings are zero–one loss

$$L_{\mathrm{id}}(y', y) = \begin{cases} 0 & \text{if } y' = y \\ 1 & \text{otherwise} \end{cases}$$

and L_{Lev}, the Levenshtein string edit distance (Wagner and Fischer, 1974; Kruskal, 1983). Both kinds of loss play a role in evaluating letter-to-sound rules: for example, Damper et al. (1999: 164) use zero–one loss, and Fisher (1999) uses string edit distance (see also Jansche, 2003).

We adopt the usual requirement that $L(y', y) = 0$ if and only if $y' = y$, which is obviously the case for L_{id}, and also holds for L_{Lev} provided the cost for matches is zero and the costs for insertions, deletions and substitutions are all nonzero, which is the case for the traditional Levenshtein distance (Kruskal, 1983). The only other condition we impose is that loss functions be computable in polynomial time, since otherwise efficient empirical risk minimization would be hopeless from the start. Zero–one loss can be computed in linear time, and edit distance in quadratic time, using a well-known dynamic programming algorithm (Wagner and Fischer, 1974). All efficiently computable metrics (in the sense of metric spaces) qualify as loss functions, though we do not require loss functions to be symmetric or to satisfy the triangle inequality. The requirement that $L(y',y) = 0$ iff $y' = y$ is crucial in our case, since it allows us to tie empirical risk minimization to the existence of consistent hypotheses: observe that $R(h) = 0$ iff h is consistent.

To summarize, the task of learning letter-to-sound rules serves as an illustration of the problem of learning local transductions, which has at its core the problem of inferring morphisms of free monoids. We focus on the restricted problem of learning so-called fine morphisms, which leads to a finite hypothesis space that is formally adequate for letter-to-sound transductions. Learning will initially be viewed as empirical risk minimization under arbitrary loss functions, as defined above.

3. Computational complexity

3.1. Optimization problem

The problem of finding a function $f : \Sigma \to \{\varepsilon\} \cup \Gamma$ for which the empirical risk $R(f^*)$ is minimal is fundamentally a combinatorial optimization problem. Like all such problems it can be stated formally in different ways (Papadimitriou and Steiglitz, 1998, p. 345f.): the *optimization version* asks for the optimal f for which the risk $R(f^*)$ is minimal on a given set of samples $D \subseteq \Sigma^* \times \Gamma^*$; the *evaluation version* asks for the average loss $R(f^*)$ incurred on D by the optimal f, but does not demand that we find that optimal f; and the *decision version* of the empirical risk minimization problem asks whether there exists an f such that the average loss incurred by it on D is less than or equal to a given budget k, that is, whether or not $R(f^*) \leq k$ for some f. A solution to the optimization version could be used to construct an answer to the evaluation version, which in turn could be used to solve the decision version. Contrapositively, if the decision version is hard to solve, so are the other two versions.

The decision version of the empirical risk minimization problem can now be stated formally, for any fixed loss function L that satisfies the criteria set out in the preceding section. The formal problem is presented in the familiar format used by Garey and Johnson (1979).

PROBLEM 1 (Fine Morphism Minimization). *Instance:* A finite (multi)set $D \subseteq \Sigma^* \times \Gamma^*$ of training samples; and a non-negative rational number k, the risk budget. *Question:* Is there a function $f : \Sigma \to \{\varepsilon\} \cup \Gamma$, corresponding to a fine morphism f^*, such that $k \cdot |D| \geq \sum_{\langle x,y\rangle \in D} L(f^*(x),y)$?

This is the most general problem we consider, but we will not discuss it much further, since its complexity may depend on the details of the loss function L. However, because we require $L(y',y) = 0$ iff $y' = y$ for all loss functions L considered here, there is a common subproblem of the decision version which is independent of the loss function used. If we set the risk budget to zero, we obtain a restricted decision version which asks whether there exists an f such that the risk $R(f^*)$ incurred on D is identically zero. We call this the *consistency problem*, since $R(f^*) = 0$ means that f^* is consistent with the training data D. Obviously, if the decision version of the optimization problem can be solved efficiently, so can the consistency problem. In other words, the formal problems considered so far can be arranged in non-decreasing order of difficulty like this: consistency problem, decision version, evaluation version, and optimization version of the optimization problem. By establishing NP hardness of the easiest of these four problems, NP hardness of the other problems follows straightforwardly.

3.2. Consistency problem

Before we can formally state the consistency problem underlying the learning task, we need another auxiliary definition:

DEFINITION 2 (Graph of a relation). Given a relation R on sets A and B, define $\#R$, the *graph of R*, to be the set $\{\langle a,b\rangle \in (A \times B) \mid aRb\}$.

Two closely related consistency problems can now be defined. An answer to the questions asked by these problem would tell us whether a suitable morphism exists that perfectly fits the training data D.

PROBLEM 2 (Very Fine Morphism Consistency – VFMC). *Instance:* A finite (multi)set $D \subseteq \Sigma^* \times \Gamma^*$ of training samples. *Question:* Does there exist a very fine morphism consistent with all elements of D, i. e., is there a function $f : \Sigma \to \Gamma$ such that $D \subseteq \#(f^*)$?

PROBLEM 3 (Fine Morphism Consistency – FMC). *Instance:* A finite (multi)-set $D \subseteq \Sigma^* \times \Gamma^*$ of training samples. *Question:* Does there exist a fine morphism consistent with all elements of D, i. e., is there a function $f : \Sigma \to \{\varepsilon\} \cup \Gamma$ such that $D \subseteq \#(f^*)$?

The size of an instance of one of these problems is the total length of all strings in the training dictionary D:

DEFINITION 3 (Dictionary size). Define the size $\|D\|$ of a dictionary $D \subseteq \Sigma^* \times \Gamma^*$ as

$$\|D\| = \sum_{\langle x,y\rangle \in D} |x| + |y|$$

where $|x|$ is the length of string x.

```
{Input: instance $D$, certificate $f$}
for all $\langle x,y \rangle \in D$ do
  $j \leftarrow 1$
  for $i \leftarrow 1$ to $|x|$ do
    if $f(x_i) \neq \varepsilon$ then
      if $j > |y|$ then
        return False
      else if $f(x_i) \neq y_j$ then
        return False
      else {$f(x_i)$ matches $y_j$}
        $j \leftarrow j+1$
  if $j \neq |y|+1$ then
    return False
return True
```

Figure 1. Certificate verification algorithm for FMC.

Of the two problems formulated here, FMC (Problem 3) is intuitively more difficult than VFMC (Problem 2), since one has to decide which input symbols are mapped to the empty string, or equivalently, how an output string should be aligned with its corresponding input string. This issue does not arise with VFMC, since only strings of equal length need to be considered (if D contains a pair of strings with different lengths, then no very fine morphism can be consistent with D). It will be shown that FMC is a complete problem for the complexity class NP (see, for example, Garey and Johnson, 1979). Membership of FMC in NP can be established straightforwardly:

THEOREM 1. *Problem FMC has succinct certificates that can be verified in polynomial time.*

Proof. A certificate for FMC is a partial function $f : \Sigma \to \{\varepsilon\} \cup \Gamma$, which can be represented in space linear in $\|D\|$ (because f only needs to mention elements of Σ that occur in D). Verification amounts to applying f^* to each input string in D and comparing the results to the corresponding reference output. The verification procedure, shown in Figure 1, runs in linear time and logarithmic space (a fixed number of counters need to be stored). □

Problem VFMC for very fine morphisms is indeed much easier than problem FMC involving fine morphisms. In fact, VFMC can be solved efficiently in linear time and space by the following procedure: for each $\langle x,y \rangle \in D$, for $i \leftarrow 1$ to $|x|$, assign (destructively) $f(x_i) \leftarrow y_i$; finally run the verification algorithm from Figure 1 on D and f, and return its answer.

NP-hardness of FMC is established by a reduction from 3SAT, the decision problem asking whether there is a satisfying truth assignment for a set (conjunction) of disjunctive clauses with at most three literals each, where a literal consists of a plain Boolean variable or of a negated Boolean variable. In other words, an instance of 3SAT is a formula ϕ in Conjunctive Normal Form with 3 literals per clause (3CNF), and the problem is to decide whether there is a satisfying truth assignment for ϕ that makes all clauses True. We first define the individual pieces (gadgets) of the reduction from 3SAT and then prove that the reduction correctly preserves the structure of 3SAT.

DEFINITION 4 (Boolean variable gadget). For any Boolean variable v, the set $\mathcal{V}(v)$ contains the following pairs (a_v and b_v are two new symbols dependent on v):

$$\langle a_v v \overline{v} b_v, FTF \rangle,$$
$$\langle a_v b_v, F \rangle.$$

The Boolean variable gadget encodes the fact that a variable occurring in a 3CNF formula can take on only the values T (True) and F (False). It consists of two entries that will become part of a larger dictionary constructed by the reduction. As will become clear below, a fine morphism that is consistent with that dictionary can map the symbols v and $\overline{v}$ only to T or F, and maps v to T iff it maps $\overline{v}$ to F. To ensure that a_v and b_v do not appear in any gadgets for other, unrelated variables, new concrete symbols a_v and b_v have to be chosen for each variable v. It is easy to see that $\|\mathcal{V}(v)\| = 10$.

DEFINITION 5 (3SAT clause gadget). For any 3SAT clause C_i of the form $(l_{i1} \vee l_{i2} \vee l_{i3})$ (where each l_{ij} is a literal of the form v or $\overline{v}$) the set $\mathcal{C}(C_i)$ contains the following pairs (c_{ij}, d_{ij}, e_i and f_i for $1 \leq j \leq 3$ are eight new symbols dependent on i):

$$\langle c_{i1}\, l_{i1}\, d_{i1}, FT \rangle,$$
$$\langle c_{i2}\, l_{i2}\, d_{i2}, FT \rangle,$$
$$\langle c_{i3}\, l_{i3}\, d_{i3}, FT \rangle,$$
$$\langle d_{i1}\, d_{i2}\, d_{i3}\, e_i\, f_i, TT \rangle.$$

The 3SAT clause gadget represents the constraint that in a clause C of the form $(l_1 \vee l_2 \vee l_3)$ at least one literal l_j must be true for the overall formula to be satisfied. We will see shortly that a fine morphism must map at least one literal l_{ij} to T and the adjacent symbol d_{ij} to ε in order for it to be consistent with the fourth pair in this dictionary. As in the Boolean variable gadget, the symbols other than those corresponding to literals or variables of the original formula must be unique for each clause, so that there are no additional constraints between clauses not present in the 3CNF formula. For each clause C we have $\|\mathcal{C}(C)\| = 22$.

DEFINITION 6 (Reduction from 3SAT). Given an instance

$$\phi = \bigwedge_{i=1}^{n} C_i$$

of 3SAT, define $\mathcal{D}(\phi)$ as the collection

$$\bigcup_{i=1}^{n} \mathcal{C}(C_i) \cup \bigcup \{\mathcal{V}(v) \mid \text{variable } v \text{ occurs in } \phi\}.$$

For example, if $\phi = (x \vee \bar{x} \vee y)$, the dictionary $\mathcal{D}(\phi)$ defined by this reduction contains the following entries:

$$\begin{array}{cc} \langle ax\bar{x}b, FTF\rangle, & \langle exf, FT\rangle, \\ \langle ab, F\rangle, & \langle g\bar{x}h, FT\rangle, \\ \langle cy\bar{y}d, FTF\rangle, & \langle iyj, FT\rangle, \\ \langle cd, F\rangle, & \langle fhjkl, TT\rangle. \end{array}$$

THEOREM 2. *The reduction from 3SAT to FMC can be computed in logarithmic space and creates an instance whose size is linear in the size of the original instance.*

Proof. The reduction $\mathcal{D}$, which can be made to run in linear time, builds a collection (dictionary) $\mathcal{D}(\phi)$ with the following properties. Let n be the number of clauses of ϕ (as in Definition 6), and let m be the number of distinct variables of ϕ (so $m \leq 3n$). Then the dictionary size (see Definition 3) is $\|\mathcal{D}(\phi)\| = 10\,m + 22\,n \leq 52\,n$, the number of pairs in the dictionary is $|\mathcal{D}(\phi)| = 2\,m + 4\,n \leq 10\,n$, the size of the input alphabet is $|\Sigma| = 4\,m + 8\,n \leq 20\,n$, and the size of the output alphabet is $|\Gamma| = 2$. Only counters need to be stored for computing the reduction (in order to keep track of clauses and variables represented by integers), which requires logarithmic space. □

We are now ready to derive the main result, which shows that deciding whether a fine morphism exists that is consistent with a given dictionary is at least as hard as any other problem in NP.

THEOREM 3. *Problem FMC is* NP-*hard.*

Proof. We show that the 3CNF formula $\phi = \bigwedge_{i=1}^{n} C_i$ is satisfiable iff there exists a fine morphism f^* consistent with $\mathcal{D}(\phi)$. It will be convenient to let V denote the set of distinct variables of ϕ.

($\Rightarrow$) Assume that ϕ is satisfiable, i. e., there exists a satisfying assignment $\tau : V \to \{T, F\}$. Incrementally define a fine morphism f^* consistent with $\mathcal{D}(\phi)$ as follows: for all $v \in V$, let $f(v) = \tau(v)$ and $f(\bar{v}) = \overline{\tau(v)}$. If $\tau(v) = T$, let $f(a_v) = F$ and $f(b_v) = \varepsilon$, which makes f^* consistent with $\mathcal{V}(v)$; otherwise, if $\tau(v) = F$, let $f(a_v) = \varepsilon$ and $f(b_v) = F$ to make f^* consistent with $\mathcal{V}(v)$. In either case f^* can be made consistent with $\mathcal{V}(v)$, and because a_v and b_v do not occur outside the gadget for v, f^* can be made consistent with all variable gadgets.

The fact that τ is a satisfying assignment means that in each clause C_i at least one literal is made True by τ. So f will map at most two d_{ij} in $\mathcal{C}(C_i)$ to T, and therefore the definition of f^* can always be extended to make it consistent with the fourth pair in $\mathcal{C}(C_i)$ and hence consistent with the entire clause gadget for C_i. Since all symbols in a clause gadget other than literals of ϕ occur only in that gadget, the definition of f^* can be extended to make it consistent with all gadgets and therefore consistent with $\mathcal{D}(\phi)$. Hence there exists a consistent fine morphism f^* constructible from τ.

($\Leftarrow$) Conversely, assume that a fine morphism g consistent with $\mathcal{D}(\phi)$ exists. Show that $g|_V$, i.e. g restricted to the variables of ϕ, is a satisfying truth assignment for ϕ. The morphism g being consistent with $\mathcal{D}(\phi)$ means that g is consistent with all variable gadgets and all clause gadgets.

Pick any variable gadget $\mathcal{V}(v)$. Then, because of the second pair in $\mathcal{V}(v)$, g must map exactly one of a_v and b_v to F: if $g(a_v) = F$ then $g(b_v) = \varepsilon$, and for the first pair $g(v) = T$ and $g(\overline{v}) = F$; otherwise if $g(b_v) = F$ then $g(a_v) = \varepsilon$, $g(v) = F$, and $g(\overline{v}) = T$. Note in particular that $(g|_V)(v) \in \{T, F\}$, so $g|_V$ is formally a truth assignment.

Now pick any clause gadget $\mathcal{C}(C_i)$ and suppose that g maps no l_{ij} in $\mathcal{C}(C_i)$ to T. Then all d_{ij} in $\mathcal{C}(C_i)$ are mapped to T because of the first three pairs in that clause gadget. But this would make g inconsistent with the fourth pair, contradicting the assumption that g is consistent with all clause gadgets. So g must map at least one l_{ij} in $\mathcal{C}(C_i)$ to T, which means that $g|_V$ makes the clause C_i True, and is therefore a satisfying truth assignment for ϕ. □

The choice of output alphabet $\Gamma = \{T, F\}$ was made merely for expository convenience. What makes FMC a hard problem is not the choice of mapping input symbols to T or F, but the choice of whether to delete an input symbol by mapping it to the empty string ε. It is in fact possible to modify the reduction so that $|\Gamma| = 1$ without changing the main theorem, though its proof would be somewhat less intuitive.

3.3. Consequences

The preceding three theorems together imply that the consistency problem FMC is NP-complete. The existence of efficient algorithms for solving FMC is therefore unlikely. In practice, it is not sufficient to know whether a consistent fine morphism exists, we also need to find such a morphism. Even if an oracle could tell us that there is at least one consistent morphism, finding one is still a difficult problem, analogous to the 3SAT function problem for satisfiable formulas, which is FNP-complete (Papadimitriou, 1994). The fact that consistency problem is hard also means that PAC learning of local transductions based on fine morphisms is impossible.

Since FMC is a special subproblem of empirical risk minimization, it follows immediately that the decision version of this optimization problem is NP-hard. It is also NP-complete, since by our earlier assumption the loss function L can be evaluated efficiently in polynomial time: a proposed solution f for the decision version can be verified efficiently by aggregating the total loss of f^* on the training data D and comparing it against the loss bound $k \cdot |D|$. Not surprisingly, the certificate verification algorithm in Figure 1 is a special case of this more general verification algorithm for the decision version of the empirical risk minimization problem.

The evaluation and optimization version of empirical risk minimization do not necessarily fall within the analogous class FNP of function problems. To show membership in FNP, we need to be able to verify the optimality of a proposed solution f in polynomial time. However, an optimal solution f only certifies the existence of a feasible solution (namely f) within a certain budget k (namely the average loss of f^* on the training data), but does not seem to provide enough information to verify in polynomial time that no better solution within a tighter budget of $k - \delta$ can exist. It is highly doubtful that there are any polynomial-length certificates of optimality. We conjecture that these problems are in fact $\mathrm{FP}^{\mathrm{NP}}$-complete, just like the Traveling Salesperson Problem TSP (see, for example, Papadimitriou, 1994).

4. Conclusions and directions for further research

We have reduced the problem of learning local transductions to the problem of learning morphisms on free monoids, after factoring out an optional preprocessing step that accumulates fixed amounts of context. The important restricted problem of deciding whether there exists a fine morphism consistent with a set of training samples was shown to be NP-complete. Since this problem is a specialization of the decision version of empirical risk minimization under many reasonable loss functions, the optimization problems which generalize the consistency problem are therefore at least as difficult.

While the main result of this paper seems discouraging, all is not lost. Given that the existence of exact efficient algorithms for solving the overall optimization problem is unlikely, one should consider the alternatives: approximate, inefficient, and/or heuristic algorithms.

4.1. Complexity of approximation

Approximate inference of consistent automata is generally a very hard problem (Pitt and Warmuth, 1993). For the special cases considered here, the complexity of approximate inference is not well understood at this point, and our initial results reported elsewhere (Jansche, 2003) in more detail are not encouraging.

Empirical risk minimization can now mean one of two things: minimizing the total number of mistakes a hypothesis makes on the training data, or maximizing the number of correct predictions on the training data. These two notions are equivalent if optimal solutions can be found exactly, but differ for approximate solutions. Say the true optimum is 10 mistakes on 100 samples, and the optimum can be approximated within a factor of 1.2: approximate maximization would find a solution with at most $100 - 90/1.2 = 25$ mistakes, but approximate minimization yields a solution with at most $10 \times 1.2 = 12$ mistakes.

Even for highly restricted problems the prospects are rather bleak. The optimization problem that asks us to maximize empirical string-level classification accuracy (the dual of empirical zero–one loss, i. e. string-level classification error) for very fine morphisms will be called MAX-VFMC. It is initially far from clear whether MAX-VFMC is an easy or a hard problem – recall that we showed in Section 3.2 that VFMC can be solved very efficiently – but it turns out (Jansche, 2003) that MAX-VFMC is APX-hard (Ausiello et al., 1999), which can be shown by a reduction from an APX-complete constraint satisfaction problem. Unless $\mathrm{P} = \mathrm{NP}$, this means that MAX-VFMC has no polynomial time approximation schemes (PTAS, which would allow us to find arbitrarily good approximations efficiently). In the best case, there may be an approximation algorithm for MAX-VFMC with a fixed approximation ratio, which would make MAX-VFMC a member of the class APX; whether or not this is the case is an open question. Similar questions regarding the complexity of related optimization problems also remain open (Jansche, 2003).

4.2. Inefficient algorithms

Exact global optimization of MAX-VFMC is theoretically possible via branch-and-bound search. This inefficient algorithm can be used for very small problem instances, e. g., learning English letter-to-sound rules with no conditioning context, for which only a few trillion morphisms have to be explored (Jansche, 2003). However, exhaustive search becomes intractable for even slightly larger problems: for English letter-to-sound rules conditioned on one letter of context there are more than one trequadragintillion feasible solutions. General heuristic algorithms, especially those based on local search (Aho et al., 1983; Papadimitriou and Steiglitz, 1998), are efficient and do in practice improve on greedily constructed initial solutions, but offer no performance guarantees. Inefficient and heuristic algorithms for solving Boolean satisfiability problems have been investigated in detail and might yield new insights on solving the formal problems presented in this paper.

4.3. Heuristics

Efficient heuristics developed specifically for learning letter-to-sound rules have been in use for at least two decades (Lucassen and Mercer, 1984). Most of them reduce the problem of learning local transductions to a classifier learning problem similar to what we have called MAX-VFMC, which involves padding the data with dummy symbols to make input and output strings equal in length. Unfortunately, this reduction does not preserve the structure of the original problem (Jansche, 2003), so that even if an optimal classifier could be found – which can be a hard problem in itself (Hyafil and Rivest, 1976) – it would not necessarily be an optimal solution for the original learning problem. It is because of this disconnect that the traditional approaches to learning letter-to-sound rules have to be seen as heuristics.

Although such approaches may seem ad hoc, there have been a number of empirical investigations (Damper et al., 1999; Bakiri and Dietterich, 2001) that find heuristic machine learning techniques preferable to knowledge representations built by human experts, though the issue remains contentious (Sproat et al., 1998). Since we concluded that the existence of sound and efficient learning algorithms for local transductions is unlikely, this can be seen as additional justification for the use of heuristics.

Acknowledgements

This paper, like an earlier version that appeared in the proceedings of MOL8, is based, occasionally verbatim, on the author's dissertation (Jansche, 2003), which contains further details. I want to thank Chris Brew, Julia Hirschberg, Gerald Penn, and the MOL8 audience and reviewers for their comments. I'd especially like to thank Richard Sproat and an anonymous JOLLI reviewer for their detailed feedback on the exposition. The usual exculpatory remarks are in effect.

References

Aho, A. V., J. E. Hopcroft, and J. D. Ullman: 1983, *Data Structures and Algorithms*, Addison-Wesley Series in Computer Science and Information Processing. Reading, MA: Addison-Wesley.

Angluin, D.: 1982, 'Inference of Reversible Languages'. *Journal of the ACM* **29**(3), 741–765.

Ausiello, G., P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi: 1999, *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Berlin, Germany: Springer.

Bakiri, G. and T. G. Dietterich: 2001, 'Constructing High-Accuracy Letter-to-Phoneme Rules with Machine Learning'. In: R. I. Damper (ed.): *Data-Driven Techniques in Speech Synthesis*, No. 9 in Telecommunications Technology and Applications. Boston, MA: Kluwer, pp. 27–44.

Damper, R. I., Y. Marchand, M. J. Adamson, and K. Gustafson: 1999, 'Evaluating the Pronunciation Component of Text-To-Speech Systems for English: A Performance Comparison of Different Approaches'. *Computer Speech and Language* **13**(2), 155–176.

Eilenberg, S.: 1974, *Automata, Languages, and Machines*, Vol. A. New York, NY: Academic Press.

Fisher, W. M.: 1999, 'A Statistical Text-To-Phone Function Using Ngrams and Rules'. In: *International Conference on Acoustics, Speech, and Signal Processing*. Phoenix, AZ, pp. 649–652.

García, P. and E. Vidal: 1990, 'Inference of *k*-Testable Languages in the Strict Sense and Application to Syntactic Pattern Recognition'. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **12**(9), 920–925.

Garey, M. R. and D. S. Johnson: 1979, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: W. H. Freeman.

Gildea, D. and D. Jurafsky: 1996, 'Learning Bias and Phonological-Rule Induction'. *Computational Linguistics* **22**(4), 497–530.

Gold, E. M.: 1967, 'Language Identification in the Limit'. *Information and Control* **10**(5), 447–474.

Hyafil, L. and R. L. Rivest: 1976, 'Constructing Optimal Binary Decision Trees is NP-Complete'. *Information Processing Letters* **5**(1), 15–17.

International Phonetic Association: 1999, *Handbook of the International Phonetic Association: A Guide to the Use of the International Phonetic Alphabet*. Cambridge, England: Cambridge University Press.

Jansche, M.: 2001, 'Re-Engineering Letter-to-Sound Rules'. In: *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*. Pittsburgh, PA, pp. 111–117.

Jansche, M.: 2003, 'Inference of String Mappings for Language Technology'. Ph.D. thesis, The Ohio State University, Columbus, OH.

Kearns, M. J., R. E. Schapire, and L. M. Sellie: 1992, 'Toward Efficient Agnostic Learning'. In: *Proceedings of the 5th Annual Workshop on Computational Learning Theory*. Philadelphia, PA, pp. 341–352.

Kearns, M. J. and U. V. Vazirani: 1994, *An Introduction to Computational Learning Theory*. Cambridge, MA: MIT Press. Second printing, 1997.

Kruskal, J. B.: 1983, 'An Overview of Sequence Comparison'. In: D. Sankoff and J. Kruskal (eds.): *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Reading, MA: Addison-Wesley, pp. 1–44. Reissued by CSLI Publications, Stanford, CA, 1999.

Lucassen, J. M. and R. L. Mercer: 1984, 'An Information Theoretic Approach to the Automatic Determination of Phonemic Baseforms'. In: *International Conference on Acoustics, Speech, and Signal Processing*. pp. 42.5.1–42.5.4.

McNaughton, R. and S. Papert: 1972, *Counter-Free Automata*. Cambridge, MA: MIT Press.

Minka, T. P.: 2000, 'Empirical Risk Minimization is an incomplete inductive principle'. http://www.stat.cmu.edu/~minka/papers/erm.html.

Mohri, M.: 1997, 'Finite-State Transducers in Language and Speech Processing'. *Computational Linguistics* **23**(2), 269–311.

Oncina, J., P. García, and E. Vidal: 1993, 'Learning Subsequential Transducers for Pattern Recognition Interpretation Tasks'. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15**(5), 448–458.

Papadimitriou, C. H.: 1994, *Computational Complexity*. Reading, MA: Addison-Wesley.

Papadimitriou, C. H. and K. Steiglitz: 1998, *Combinatorial Optimization: Algorithms and Complexity*. Mineola, NY: Dover Publications. Originally published by Prentice Hall, Englewood Cliffs, NJ, 1982.

Pitt, L.: 1989, 'Inductive Inference, DFAs, and Computational Complexity'. In: K. P. Jantke (ed.): *Analogical and Inductive Inference, International Workshop AII '89, Reinhards-*

brunn Castle, GDR, October 1–6, 1989, Proceedings, Vol. 397 of *Lecture Notes in Computer Science*. Berlin, Germany, pp. 18–44, Springer.

Pitt, L. and M. K. Warmuth: 1993, 'The Minimum Consistent DFA Problem cannot be Approximated within Any Polynomial'. *Journal of the ACM* **40**(1), 95–142.

Roche, E. and Y. Schabes (eds.): 1997, *Finite-State Language Processing*, Language, Speech and Communication. Cambridge, MA: MIT Press.

Sejnowski, T. J. and C. R. Rosenberg: 1987, 'Parallel Networks that Learn to Pronounce English Text'. *Complex Systems* **1**(1), 145–168.

Sproat, R., B. Möbius, K. Maeda, and E. Tzoukermann: 1998, 'Multilingual Text Analysis'. In: R. Sproat (ed.): *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach*. Dordrecht, The Netherlands: Kluwer, Chapt. 3, pp. 31–87.

Valiant, L. G.: 1984, 'A Theory of the Learnable'. *Communications of the ACM* **27**(11), 1134–1142.

van den Bosch, A. P. J.: 1997, 'Learning to Pronounce Written Words: A Study in Inductive Language Learning'. Ph.D. thesis, Universiteit Maastricht, Maastricht, The Netherlands.

Wagner, R. A. and M. J. Fischer: 1974, 'The String-to-String Correction Problem'. *Journal of the ACM* **21**(1), 168–173.

Weide, R. L.: 1998, 'The Carnegie Mellon Pronouncing Dictionary Version 0.6'. Electronic document, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. ftp://ftp.cs.cmu.edu/project/fgdata/dict/.

Address for Offprints:
Martin Jansche
Center for Computational Learning Systems
Columbia University
500 W 120th St, MC 4750
New York, NY 10027
USA