Learning Local Transductions is Hard

Martin Jansche
The Ohio State University

MoL 8
June 21, 2003

Introduction

- Typical application: so-called letter-to-sound rules.
- The task of predicting the pronunciation of a word on the basis of its spelling, without looking it up in a dictionary.
- Long history of machine learning of letter-to-sound rules, dating back to the early 1980s.
- Most approaches try to formulate this as a classification: label each letter with its pronunciation.
- Learning letter-to-sound rules turns into classifier learning.

How is this different from other sequence learning problems?

- Sequence learning seems to be well understood [Dietterich 2002 in LNCS 2396].
- Has been applied to various phrase labeling tasks, including part-of-speech tagging, word sense tagging, named entity recognition, noun phrase chunking, sentence boundary detection, restoration of punctuation, restoration of accent diacritics, etc.
- All those tasks are essentially same-length transductions.

What's so special about this task?

Look at some data:

- ⟨featherweight⟩ 13 letters
  /f eh dh er w ey t/ 7 phonemes

- ⟨mutualism⟩ 9 letters
  /m y uw ch ax w ax l ih z ax m/ 12 phonemes

- ⟨parliamentarianism⟩ 18 letters
  /p aa r l ax m ax n t eh r iy ax n ih z ax m/ 18 phonemes

But can't we at least pretend they have the same length?

Sure, but not naively:

- ⟨f e a t h e r w e i g h t⟩
  /f eh dh er w ey t – – – – – – –/

- ⟨m u t u a l i s m bork bork bork⟩
  /m y uw ch ax w ax l ih z ax m /

- ⟨p a r l i a m e n t a r i a n i s m⟩
  /p aa r l ax m ax n t eh r iy ax n ih z ax m/

That's an ugly hack, fix it!

- ⟨f e a t h e r w e i g h t⟩
  /f eh – dh – er – w ey – – – t/

- ⟨m – u t – u a l i s – m⟩
  /m y uw ch ax w ax l ih z ax m/

- ⟨m u t u a l i s m ⟩
  /m y+uw ch+ax w ax l ih z ax+m/

- It's still a hack (though less ugly): Can this transformation be automated? How do we tell if it's any good?

In Familiar Territory

- Now we can view learning letter-to-sound rules as learning same-length rational relations.
- The traditional machine learning approaches [e.g. Sejnowski and Rosenberg 1987] effectively restrict the problem even further.
- Local letter context provides the most useful features for classification [Lucassen and Mercer 1984].
- Learning deterministic local same-length transductions.

Deterministic Local Transductions

Computed by scanner machines, analogous to locally testable languages in the strict sense [McNaughton and Papert 1972].

s l a u g h t e r h o u s e

#sl sla lau aug ugh ght hte ter erh rho hou ous use se#

s l ao – – – t – er h aw – s –

Morphisms of Free Monoids

- Accumulating local context is a preprocessing step.
- The intermediate sequence of local windows is viewed as a string over a new, larger (but still finite) alphabet.
- Need to learn a function from local windows to labels. This gives rise to an alphabetic substitution or so-called *very fine* morphism [Eilenberg 1974].
- A very fine morphism is a monoid morphism of free monoids that is uniquely determined by a function $f : \Sigma \to \Gamma$. Such a function can be uniquely extended to $f^* : \Sigma^* \to \Gamma^*$.

The Learning Tasks

- Generalize over a pronunciation dictionary.
- Preprocess the letter strings deterministically to insert the desired amount of local letter context.
- Two choices at this point:
  - Ensure that the training data are pairs of same-length strings (**aligned**). Learn a very fine morphism.
  - Try to discover alignments between letters and phonemes automatically during learning. Learn a *fine* morphism f^* with $f : \Sigma \rightarrow \Gamma \cup \{\epsilon\}$.

Conceptualization of Learning

- Identification in the limit [Gold 1967]? Too unrealistic.
- Probably approximately correct (PAC) learning [Valiant 1984]? A worthwhile goal, but complicated. Key ingredients:
  - Determine the size of the hypothesis space.
  - Formulate efficient algorithms that output consistent hypotheses.
- Empirical risk minimization. Often used in practice, some relations to PAC learning.

Empirical Risk Minimization

- Risk is expected loss on all data, including future unseen data.
- Loss is a function into the nonnegative reals. Often satisfies additional conditions. Many loss functions are metrics.
- Empirical risk is average loss on training data. An estimate of risk.
- Minimizing empirical risk is the same as minimizing loss summed over the training data.

Loss Functions

- A loss function compares predicted phoneme strings with a recorded gold standard and quantifies the difference.
- Discrete 0-1-loss: zero loss for entirely correct phoneme string, unit loss for phoneme string with any error.
- String edit distance [Wagner and Fisher 1974].
- Any other function that maps identical strings to zero loss and nonidentical strings to nonzero loss (all metrics qualify).
- Common trait: A hypothesis is consistent iff it has zero loss.

The Main Learning Task

- Given a set of training samples, find a fine morphism with minimal loss.
- This is a combinatorial optimization problem [Papdimitriou and Steiglitz 1982].
- Every optimization problem has an associated decision problem. If the optimization problem can be solved efficiently, so too can the decision problem. Contrapositively, if the decision problem cannot be solved efficiently, neither can the optimization problem.

The Main Decision Problem MIN-FMC

Problem instance:

- a finite multiset $D = \{\langle x_1, y_1\rangle, \ldots, \langle x_n, y_n\rangle\}$ of string pairs;
- a nonnegative rational number k (the loss bound).

Question (for a fixed loss function $L : \Gamma^* \times \Gamma^* \to \mathbf{Q}_+$):

Is there a function $f : \Sigma \to \Gamma \cup \{\epsilon\}$, corresponding to a fine morphism f^*, such that $\sum_{\langle x,y\rangle \in D} L(f^*(x), y) \leq k$?

A Useful Simplification

- Set the budget k to zero.
- This asks whether there is a hypothesis f^* with zero loss, i.e. a consistent hypothesis.
- By our earlier assumption, zero loss means all phoneme strings y_i in the training dictionary D are correctly predicted by f^*.
- We don't need to mention the loss function L at all.
- Ties in with PAC learning.

The Consistency Problem FMC

Problem instance:

A finite multiset $D = \{\langle x_1, y_1\rangle, \ldots, \langle x_n, y_n\rangle\}$ of string pairs.

Question:

Is there a function $f : \Sigma \to \Gamma \cup \{\epsilon\}$, corresponding to a fine morphism f^*, such that $f^*(x) = y$ for all $\langle x, y\rangle \in D$?

FMC is NP-complete

- Show two things: membership in NP and NP-hardness.
- Membership is very easy. A nondeterministic Turing machine could simply guess a function f and then deterministically check whether f^* is consistent with the training data D.
- We show NP-hardness by a reduction from an NP-complete problem. 3SAT is the problem of deciding whether a Boolean formula in conjunctive normal form with at most three literals per clause (3CNF) has a satisfying truth assignment (or whether it is a contradiction).

Reducing 3SAT to FMC

- A 3CNF formula ϕ is a conjunction $\bigwedge_i C_i$ of clauses. Each clause C_i is of the form $(l_{i1} \vee l_{i2} \vee l_{i3})$, i.e. a disjunction of three literals. A literal is either a positive or negated variable.
- 3SAT is the problem of deciding for a given formula ϕ whether it is satisfiable.
- A reduction maps a 3SAT instance ϕ to an FMC instance D while preserving the structure of the satisfiability problem.
- Reduction broken down into "gadgets".

The Boolean Variable Gadget

- Encodes the fact the a Boolean variable assumes the values T or F and that v is False iff its negation is True.
- A variable v is mapped to a dictionary with the following two members:

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

$$\langle a_v b_v,\ F \rangle$$

- There are exactly two consistent fine morphisms g and h: $g(v) = T$, $g(\overline{v}) = F$; and $h(v) = F$, $h(\overline{v}) = T$.

The 3CNF Clause Gadget

- Encodes the fact that a clause is satisfied iff at least one of its literals is true.
- A clause $C_i = (l \vee m \vee n)$ is mapped to a dictionary with the following four members:

$$\langle p_i\, l\, q_i,\ FT \rangle$$

$$\langle r_i\, m\, s_i,\ FT \rangle$$

$$\langle t_i\, n\, u_i,\ FT \rangle$$

$$\langle q_i\, s_i\, u_i\, v_i\, w_i,\ TT \rangle$$

The 3CNF Clause Gadget

- At most two symbols among q_i, s_i and u_i can be mapped to T by a consistent fine morphism. At least one symbols must be mapped to the empty string ϵ. But that means the corresponding literal is mapped to T, so the clause is satisfied.

- The converse also holds: If C can be satisfied, then there exists a fine morphism that is consistent with all four elements of the gadget.

The Reduction

- Given a 3CNF formula $\phi = \bigwedge_i C_i$, let V be the set of variables occurring in ϕ. Let $\mathcal{V}(v)$ be the variable gadget for $v \in V$, and let $\mathcal{C}(C_i)$ be the clause gadget for clause C_i.
- Construct the dictionary $\mathcal{D}(\phi) = \bigcup_{v \in V} \mathcal{V}(v) \cup \bigcup_i \mathcal{C}(C_i)$.
- First claim: This reduction can be carried out in time polynomial in the size of ϕ (stronger: in logarithmic space).
- Second claim: ϕ is satisfiable iff there is a fine morphism consistent with $\mathcal{D}(\phi)$.

What does this tell us?

Unless P=NP:

- There are no efficient algorithms for finding a consistent fine morphism. If there was such an algorithm, it could be used to solve any problem in NP efficiently, by reducing it to FMC (via 3SAT) and then running the hypothetical algorithm.
- But FMC is a subproblem of MIN-FMC. So there cannot be efficient algorithms for MIN-FMC either.
- But the decision problem MIN-FMC is no harder than the corresponding optimization problem.

Conclusions

- Learning local transductions from unaligned data can be viewed as inference of fine morphisms.
- Deciding whether a fine morphism consistent with a given training set exists is a hard problem, and therefore finding such a morphism is hard too.
- Empirical risk minimization under several commonly used loss functions (discrete loss, string edit distance) is hard too.
- Research issue: Can the optimization problem be approximated efficiently?

Evaluation Criteria in Action

	s	l	a	u	g	h	t	e	r	h	o	u	s	e	*Cls*	*Str*	*Sym*
Ref	s	l	ao	–	–	–	t	–	er	h	aw	–	s	–			
#1	–	–	–	s	l	ao	–	t	–	–	er	h	aw	s	$\frac{14}{14}$	$\frac{0}{1}$	$\frac{0}{8}$
#2	s	l	ao	–	–	–	t	–	er	h	aw	–	z	–	$\frac{1}{14}$	$\frac{1}{1}$	$\frac{1}{8}$
#3	t	ow	t	ll	ah	t	er	jh	ih	b	ax	r	ih	sh	$\frac{14}{14}$	$\frac{1}{1}$	$\frac{12}{8}$

	Reference	*Predicted*	*Cls*	*Sym*	*Str*
flexure	`flEK-R-`	`fl-z-r-`	3		
	f l eh k sh er	f l z r		4	1
inflexion	`InflEK-xn`	`Infl-zIxn`	3		
	ih n f l eh k sh ax n	ih n f l z ih ax n		3	1
lynx	`lIGX`	`lAnz`	3		
	l ih ng k s	l ay n z		4	1
prefix	`prifIX`	`fr-fIz`	3		
	p r iy f ih k s	f r f ih z		4	1
xenophobe	`zEnxf-ob-`	`z-nxf-xb-`	2		
	z eh n ax f ow b	z n ax f ax b		2	1
xerophyte	`zIrxf-At-`	`z-rxf-At-`	1		
	z ih r ax f ay t	z r ax f ay t		1	1
xylophone	`zAlxf-on-`	`zAlxf-xn-`	1		
	z ay l ax f ow n	z ay l ax f ax n		1	1
		Totals	**16**	19	7

	Reference	*Predicted*	*Cls*	*Sym*	*Str*
flexure	`flEK-R-`	`fl-k-r-`	3		
	f l eh k sh er	f l k r		3	1
inflexion	`InflEK-xn`	`Infl-kIxn`	3		
	ih n f l eh k sh ax n	ih n f l k ih ax n		2	1
lynx	`lIGX`	`lAnk`	3		
	l ih ng k s	l ay n k		3	1
prefix	`prifIX`	`fr-fIk`	3		
	p r iy f ih k s	f r f ih k		3	1
xenophobe	`zEnxf-ob-`	`k-nxf-xb-`	3		
	z eh n ax f ow b	k n ax f ax b		3	1
xerophyte	`zIrxf-At-`	`k-rxf-At-`	2		
	z ih r ax f ay t	k r ax f ay t		2	1
xylophone	`zAlxf-on-`	`kAlxf-xn-`	2		
	z ay l ax f ow n	k ay l ax f ax n		2	1
		Totals	19	**18**	7

	Reference	*Predicted*	*Cls*	*Sym*	*Str*
flexure	`flEK-R-`	`fliX-ri`	4		
	f l eh k sh er	f l iy k s r iy		4	1
inflexion	`InflEK-xn`	`IGfliXIxG`	5		
	ih n f l eh k sh ax n	ih n f l iy k s ih ax ng		5	1
lynx	`lIGX`	`lIGX`	0		
	l ih ng k s	l ih ng k s		0	0
prefix	`prifIX`	`prifIX`	0		
	p r iy f ih k s	p r iy f ih k s		0	0
xenophobe	`zEnxf-ob-`	`XiGxp-xbi`	6		
	z eh n ax f ow b	k s iy ng ax p ax b iy		7	1
xerophyte	`zIrxf-At-`	`Xirxp-Iti`	5		
	z ih r ax f ay t	k s iy r ax p ih t iy		6	1
xylophone	`zAlxf-on-`	`XIlxp-xGi`	6		
	z ay l ax f ow n	k s ih l ax p ax ng iy		7	1
		Totals	26	29	**5**