Learnability Theory
(a huge topic in its own right! I’ve adapted this material from Ed Stabler’s learnability course notes at UCLA and from the Jain & al. chapter)

(1) Learnability Theory can seek to answer mathematical questions like…

- Can a given language or type of language (where language = set of strings (“expressions”)) be learned from positive evidence only? From positive and negative evidence? Using how powerful an algorithm?
- How long does it take?
- When does the learner know for sure that the language has been successfully learned?

(2) Gold Paradigm

- A language is a (possibly infinite) set of symbols. In the Jain & al. chapter, each symbol is a natural number. We can think of each natural number as the index for an utterance.
  (How? Express every possible utterance as a string, perhaps using nested brackets to indicate hierarchical structure, put the strings in alphabetical order, and number them 0, 1, 2, ...)

- The learner is a function that takes a finite sequence of symbols and returns a grammar (=a definition of a language). In Jain & al., it actually produces the index of a grammar, where the set of all possible grammars is alphabetized similarly to the set of all possible utterances.

- The learner is exposed to a text (=infinite sequence containing all and only the symbols that are members of the target language, possibly with repetitions).

After being exposed to the first symbol in the sequence, the learner returns some grammar (often the wrong one), or perhaps returns nothing. After being exposed to the next symbol, the learner may produce the same grammar, or a different one.

- If the learner eventually keeps producing the same grammar and will never change to a different grammar no matter what strings it sees later, the learner has converged to that grammar on the text.
• If that converged-to grammar is correct (produces a language containing exactly
the symbols that the text contains, although the learner may not yet have seen all
of the symbols), the learner has identified the text.

• If the learner can identify any text for some language, the learner identifies that
language.

• If a learner can identify any language in some class of languages, the learner identifies that class of languages.

• If a class of languages is identified by some learner, that class is identifiable (i.e.,
learnable).

(3) \textit{Theorem (Gold 1967)}
The class of all finite languages is identifiable.
(Let’s prove this)

(4) \textit{Self-awareness}
In language learning, we probably shouldn’t require the learner to know when it’s
succeeded (except if we require linearizing the stratal grammar as the last step of
CD).

Learners that know when they’ve converged are self-monitoring.

Self-monitoring-ness is not too important in the Gold paradigm. For instance,
there is no self-monitoring learner that identifies the class of all finite languages.

(5) \textit{Computability}
We haven’t guaranteed that there’s a way of checking if a given grammar
generates a given set of sentences. So a class of languages may be identifiable, but
not computably so.

(6) \textit{Learnability of “constraint-based” grammars}
Simple case: \(*x\).
The set of languages that obeys \(*x\) is \(\Sigma^* - \{x\} \mid x \in \Sigma\).
This set is learnable (let’s prove it).

Similarly, the class of languages that forbid 2 strings, or the class of languages
that forbid 3 strings, etc., is learnable. The learner just has to know how many
forbidden strings to expect.

But the class of languages that forbid either 1 or 2 strings is not identifiable.

(7) \textit{Locking sequences}
Theorem: If a learner identifies some language, there exists is a sequence \(\sigma\) of
symbols that could begin a text for that language, and given that initial sequence,
the learner will give the correct grammar and never diverge from it no matter what other symbols from the language it sees later. $\sigma$ is a locking sequence.

We could also think about sequences that cause a learner to produce some incorrect grammar and never diverge from it.

(8) Memory limitation
What if the learner can only ‘see’ the current grammar and next symbol? Then the learner is memory-limited, and there are some identifiable language classes that it can’t identify.

CD and GLA both work like this.

(9) Applications to OT?
If there is a finite set of universal constraints, then although each language is infinite, the set of possible languages is finite.

Any finite class of languages is identifiable, so the problem is about how long it takes, not whether it can be done in principle.

**Probably Approximately Correct (PAC) learning**
This is a more appropriate paradigm for looking at the GLA.

(10) Standard PAC learning
- Call the hypothesis space $C$.
- The possible hypotheses are sets.
- The learner’s goal is to figure out the definition of some set $C \in C$ based on a text.
- The text is a random sequence of elements of a set $\Omega$; their frequencies may be skewed.
- Each datum is labeled as belonging to $C$ or not.
- $C$ is PAC-learnable if there exists some learner that, given enough of the text, can generate a hypothesis about the definition of $C$ that is correct to within whatever margin of error you select, with whatever confidence level you select.
- For example, you might want a learner that you can be at least 95% certain has identified $C$ with at least 90% accuracy.
• Accuracy is measured as how often the definition the learner came up with will correctly categorize a new item as belonging to \( C \) or not.

• The main issue is how big the training set has to be to get the desired accuracy with the desired probability. If the time needed is polynomial on the accuracy and probability and the size of the hypothesis space \( C \), then \( C \) is efficiently PAC-learnable.

(11) **PAC-learning modified for language**

• Call the hypothesis space \( C \).

• The possible hypotheses are languages.

• The learner’s goal is to figure out the grammar for some language \( C \in C \) based on a text.

• The text is a random sequence of elements of a set \( \Omega \); their frequencies may be skewed. (Should elements be just utterances, or utterances together with the input?)

• Only data that belong to \( C \) are presented (no negative evidence).

• \( C \) is **PAC-learnable** if there exists some learner that, given enough of the text, can generate a hypothesis about the definition of \( C \) that is correct to within whatever margin of error you select, with whatever confidence level you select.

• Accuracy is measured as...how often the grammar the learner came up with will correctly categorize a new utterance as belonging to \( C \) or not? How often the grammar will produce the correct output for some input? How often the grammar will recognize the correct input for some output?

(12) **How to do PAC learning of OT?**

• If the \( C \) to be identified is defined by a constraint ranking, we could express that as a set of pairwise rankings (\( A >> B, C >> D, D >> A \), etc.).

• For each pair of constraints, the learner has to decide if \( C1 >> C2 \) is true or false.

• Sets whose members are true/false choices for a list of propositions (“monomials”) are efficiently PAC-learnable, but that’s assuming negative evidence.

Nice web page on PAC-learning of concepts:
http://yoda.cis.temple.edu:8080/UGAIWWW/lectures95/learn/pac/pac.html
Learnability issues in OT

(13) **Problem of negative evidence**
If you never hear some structure in your language, and that structure is not ruled out on universal grounds, how do you know that the structure is forbidden in your language, and not just accidentally absent from your experience so far?  
(By the way, in statistical approaches to learning, this needn’t be a problem: the longer you go without hearing it, the surer you are that it’s forbidden.)

If I’m learning Tahitian, how do I know that *CODA >> FAITH?*

Prince’s solution: if markedness constraints start out at the top and are preferentially kept there, *CODA >> FAITH until evidence to the contrary is encountered.*

(14) **Learning inputs**

**Lexicon Optimization:** pick the input that, mapped to the output in question, is most harmonic. This will be the input that is most similar to the output.

What about alternations? There, you need to construct an input that may be similar to some outputs, but is different from others. How do you decide which output to be similar to?

If you already have the grammar, Tesar & Smolensky propose Lexicon Optimization over the paradigm as in 5.2, for German-type coda devoicing.

But this is not generally how we think about the German case! /tag/ is thought to be underlyingly voiced not because this saves a violation of **ONSFAITH**, but because the words that don’t alternate are always voiceless.

Let’s see how this would work if we had a language with intervocalic voicing rather than coda devoicing.

**Next Time**
(next Tuesday is a holiday; next class is Oct. 16)

- Comprehension

**For next time**
- Read: Tesar & Smolensky (2000) ch. 4
  You might want to get ahead by reading Samek-Lodovici & Prince (1999) too.
- Homework: Try doing the procedure on p. 82 of Tesar & Smolensky on some data, perhaps one you used in a previous homework. Do robust interpretive parsing, then constraint demotion, then paradigm lexicon optimization, and repeat from the top. (For robust interpretive parsing, you’ll have to read ch. 4). Show all steps, and if you run into any problems or the result is wrong, discuss. If you prefer, you can do GLA instead of CD for the middle step.