1 Summary of Part 1

1.1 Computational characterizations of linguistic generalizations

1. It is important to be able to relate the intensional description of a linguistic generalization (a grammar) to its extensional description (a potentially infinite set of points).
2. This requires some mathematics.
3. Formal grammars like automata and logic are well-studied tools that accomplish this.
4. They provide insights into the nature of natural language patterns not obtainable in other ways.
5. They are not finished! There is a lot yet to accomplish to develop formal grammars for linguistics, and we should not ignore the lessons of previous research.
6. For linguistics:
   - Well-formedness can be characterized with *sets*
   - Transformations can be characterized with *functions*
7. So what is the nature of these sets and functions for linguistics?

1.2 Algorithms

1. Algorithms are procedures which solve well-defined problems after finitely many steps.
2. Proving an algorithm solves a well-defined problem is important.
3. Proving an algorithm finds the answer to any instance of the problem with a certain amount of resources is also important.
4. This also requires some mathematics.
5. These results guarantee the *general* behavior of the algorithm.
6. This stands in contrast to simulations, which only show a program's specific behavior on a specific instance of some problem.
7. Problems for linguistics where algorithms help:
   - Membership problems
   - Transformation problems
   - Learning problems.

1.3 Defining Learning

1. What is a reasonable definition? Many issues and answers.
   - When is the data is good enough?
   - What counts as successful learning?
2. Learning problems ought to be defined for classes of generalizations, not individual generalizations.

1.4 Learning Definitions

1. Identification in the limit

   (a) Does the learner only make finitely many mistakes for any of the allowable presentations of examples?
      - from positive data on arbitrary presentations
      - from positive and negative data on arbitrary presentations
      - from positive data on recursive presentations
      - from positive data on primitive recursive presentations
2. Maximum likelihood estimate for stochastic sets
   (a) Does the learner do its best? (This means any deviation from its output worse.)
      - It is constrained by the data it gets.
      - It is constrained by its hypothesis space/structural limitations/parametric model/space of parametric models.
3. Other definitions we did get to.
   (a) Probably Approximately Correct learning
   (b) Maximizing the margin (for classification)
   (c) Maximizing the entropy (related to MLE)
   (d) Stochastic finite learning
   (e) ...
4. Complexity issues we did not get to.
   (a) Mistake bounds
   (b) VC dimension
   (c) Update-time and "Pitt's trick"
   (d) ...

1.5 Important results in learning theory

1. Finite class of stringsets is identifiable in the limit from positive data on arbitrary presentations.
2. No superfinite class of stringsets is identifiable in the limit from positive data on arbitrary presentations.
3. The computably enumerable stringsets are identifiable in the limit from positive and negative data on arbitrary presentations. (This learner by enumeration is not efficient.)
4. The computable stringsets are identifiable in the limit from positive data on primitive recursive presentations. (This learner by enumeration is not efficient.)
5. Gold's conclusions and critical analysis/reflection.
   - Maybe not all context-free (context-sensitive) stringsets are possible human languages.
   - Maybe humans access negative evidence in some way.
   - Maybe the data humans receive is not arbitrary in some way.
6. Results we did not go over:
   (a) The regular class of stringsets is efficiently identifiable in the limit from positive and negative data (RPNI).
   (b) Deterministic regular transductions are identifiable in the limit from positive data

(OSTIA).

(c) Deterministic regular probability distributions are identifiable in the limit from positive data (RLIPS, ALEGRIA).

(d) Corresponding results extend these to tree languages and tree transductions.

1.6 String extension learning and automata learning

1. String extension learning
   (a) Basic idea: Well-formedness of a structure is determined by its "parts"
   (b) Formal grammars are finite sets.
      - substrings (SL)
      - subsequences (SP)
      - substrings on a tier (TSL)
      - sets of substrings (LT)
      - sets of subsequences (PT)
      - multisets of substrings (LTT)
      - local trees (SL treesets)
      - lots of possibilities ...
   (c) There is a function which maps structures like strings or trees to sets.
   (d) The formal grammar defines a set of structures (like strings or trees) as all structures whose image under the function is a subset of the grammar.
   (e) Learning builds a grammar by unioning the images of the examples under the function. (It begins with the empty set.)
   (f) Identification in the limit from positive data.
2. Automata learning stringsets
   (a) Basic idea: Memory required to determine well-formedness is independent of length of the input.
   (b) A finite-state machine is a grammar solving some membership or transformation problem.
   (c) Deterministic finite acceptors (DFAs) correspond to classes of stringsets.
      - Each SL_k stringset is a sub-graph of the SL_k DFA.
      - Each $\mathrm{TSL}_{T,k}$ stringset is a sub-graph of the $\mathrm{TSL}_{T,k}$ DFA.
      - Each LT_k stringset is a sub-graph of the LT_k DFA.
      - Each PT_k stringset is a sub-graph of the PT_k DFA.
      - Each $\mathrm{LTT}_{t,k}$ stringset is a sub-graph of the $\mathrm{LTT}_{t,k}$ DFA.
   (d) Learning simply traces the path of the sample in the DFA.
3. Automata learning stochastic stringsets
   (a) Probabilities are added to the transitions.
   (b) DFAs correspond to classes of stochastic stringsets as before.
   (c) Learning simply counts the paths of the sample in the DFA and normalizes.
   (d) Provably gets the MLE.
4. Automata learning string-to-string functions

   (a) Output strings are added to the transitions.
   (b) DFAs correspond to classes of string-to-string transformations as before.
   (c) Learning assigns to each transition the contribution (minimal change) each symbol makes at state q by factoring out the common output of all input strings which lead to q.
   (d) Provably efficiently learns the class of transductions.
5. Work in progress
   (a) SP_k is characterized by a *set* of DFAs and learning traces the sample through each DFA. (The stringset the set of DFAs defines is given by intersection.)
   (b) Heinz and Rogers (2010) generalized this idea to learn the MLE of stochastic SP_k stringsets.
   (c) Shibata and Heinz (in prep) provide a much better proof of this result, and generalize (I think) to sets of DFAs under some conditions.
   (d) The transducer case is wide open, though I have some ideas.

1.7 Open Questions

1. ISL and OSL functions are *functions*—so each input has at most one output.
   (a) How can we add variation to this?
   (b) How can we add probabilities to this?
   (c) Once accomplished, this has MANY potential applications in NLP.
2. The proper notion of k-factor and generalizing these results to representations
   (a) Subsequences are k-factors with the right representations
   (b) SL tree languages are k-factors with the right representations
   (c) What about k-factors of autosegmental structures?
   (d) What about k-factors for feature-based representations?
3. The subregular classes for strings and string-to-string functions can be lifted to trees.
   (a) SP, LT, PT, TSL treesets? What are these and do they capture syntactic generalizations?
   (b) ISL and OSL tree transductions?
4. Expanding the grammatical architecture
   (a) We have examined learning generalizations within individual modules of grammar (phonotactics, phonological transformations, morphological transformations, syntactic well-formedness).
   (b) How can more than one component be learned simultaneously?
   (c) What is the role or character of the lexicon in morpho-phonological learning?
   (d) How can learning be defined in the face of exceptions? (cf. Tolerance Principle)
   (e) What algorithms can “successfully learn” in the face of exceptions?
5. For all these questions, it is imperative to define a learning problem. What are the instances of the problem (example data)? What are its answers (target grammars)?