Research seminar week 5

Tamás Biró

Humanities Computing

University of Groningen

`t.s.biro@rug.nl`

This week: learning algorithms

- Online learning: error driven learning
  - TLA for P&P
  - EDCD and GLA for OT (and HG)
- Offline learning:
  - RCD for OT (and HG)
- Iterated learning

Error driven learning

```
GENERAL EROR DRIVEN LEARNING ALGORITHM
  Input: H_0 starting hypothesis; learning data set
  H <-- H_0
  Repeat read w from data set
      If w not in language generated by H
          then change H to some (good/better) hypothesis
  Until no more change is needed
  Return H
```

Triggering Learning Algorithm (TLA) for P&P

- "Hill climbing 2"-type of learning. Memoryless.
- If w not in H: select one parameter at random, and flip it. If w in new grammar, then change H to it.

Triggering Learning Algorithm (TLA) for P&P

- Local optima.

- Alternatives: change more than one parameter; always move (no greediness). Niyogi 4.2: improves TLA.

Learning in OT (and HG)

- Observed form (winner) vs. form generated by current hypothesis hierarchy (loser).

- Demote constraints violated by winner and not by loser below at least one constraint violated by loser and not by winner.

Learning in OT (and HG)

- Online: Error Driven Constraint Demotion (EDCD; by Tesar)
- Online + stochastic OT: Gradual Learning Algorithm (GLA; by Boersma)
- Offline: Recursive Constraint Demotion (RCD; by Tesar)
- (and many other, more recent variants...)

Basic idea of learning in OT

Winner form w observed, loser form l produced by current ranking:

	...	C_1	C_2	...	C_k	...
w		2	3		1	...
l		1	0		3	...

- Ignore constraints C_i s. t. $C_i(w) = C_(l)$.

Basic idea of learning in OT

- l wins for this hierarchy because l has less violations than w at the highest constraint C_i such that $C_i(w) \neq C_i(l)$.

- In order to get a hierarchy in which w wins to l, **all** constraints for which $C_i(w) > C_i(l)$ must be lower ranked than at least **one** constraint for which $C_i(w) < C_i(l)$.

EDCD: Error Driven Constraint Demotion (by B. Tesar)

- In each step, if actual hypothesis hierarchy produces form l different from observed data w, then
  - find highest ranked constraint C_k such that $C_k(w) < C_k(l)$;

– find all constraints C_i such that $C_i(w) > C_i(l)$ and C_i is currently ranked higher than C_k;

– demote all the latter ones below C_k.

- Algorithm gets stuck if errors in data.

- For details (which should not necessarily be followed), such as the idea of strata, refer to *Tesar and Smolensky 2000*.

GLA: Gradual Learning Algorithm (by P. Boersma)

- Each constraint C_i is assigned a $rank$, that is, a real number r_i. A higher rank means a higher position in the hierarchy.

- In each step, if actual hypothesis hierarchy produces form l different from observed data w, then

  – find all constraints C_k s. t. $C_k(w) < C_k(l)$;

increase their rank r_i by a small number p ("plasticity");

– find all constraints C_i such that $C_i(w) > C_i(l)$; and decrease their rank r_i by a small number p ("plasticity").

- Algorithm is robust to small percentage of errors.

- For details, such as how plasticity can be used and the use of this learning algorithm for $Stochastic$ OT, refer to Boersma and Hayes 2001.

RCD: Recursive Constraint Demotion (by B. Tesar)

- Collect all winner forms. Compare them to all their losing competitors.

- Create a table: for each (winner w, loser l) pair: winner marks (constraints such that $C(w) > C(l)$) vs. loser marks (constraints such that $C(w) < C(l)$).

- Build hierarchy from the top:
  1. Add constraint C to hierarchy if C never appears in the table as winner mark.
  2. Remove rows from table where C appears as loser mark.
  3. Go back to step 1 if table is not empty.

- Algorithm detects errors and stops (table not empty, and yet all constraints appear somewhere as winner mark).

A note on HG

- You can employ same algorithms as in OT: work with hierarchies.

- Exponential weights: assign weight -1 to lowest ranked constraint, $-q$ to second lowest ranked constraint, $-q^2$ to third lowest ranked constraint, etc. ($q > 1$; test different q values, such as 2, 10, etc.)

By next week:

- Your presentations