Automatic Learning with Feedback Queries

Automatic Refers to Accepted by Finite Automata

John Case[1]
Sanjay Jain[2]
Yuh Shin Ong[2]
Pavel Semukhin[3]
Frank Stephan[2]

[1]University of Delaware

[2]National University of Singapore

[3]University of Regina

Computability in Europe 2011
Sofia, Bulgaria

For Your Speed Reading Pleasure & Quick Impression (☺)

1. Background
   - Motivation & Numerical Example
   - Semi Computability-Theoretic Setting
   - Learnable Classes of Regular Languages
2. Automatic Structures & Learning
   - Automatic Structures
   - Automatic Classes
   - Learning Automatic Classes & Further Motivation
   - Memory Restrictions
   - Formulate Automatic Feedback Learning
3. Examples & Results
   - Examples
   - Results

Motivation & Numerical Example

- A motivation: Program of Khoussainov & Nerode 1995 re $\approx$ effect of replacing TMs by finite automata in computable model theory.
- Present paper: One in a series (Jain, Luo and Stephan 2010; Jain, Ong, Pu, Stephan 2010; Case, Jain, Le, Ong, Semukhin, Stephan LATA-2011) devoted to effect on computability-theoretic learning theory under above replacement.
- Example unrestricted learning from positive data:

Data	Hypotheses
2	Set of all even numbers;
2,3	Set of all numbers;
2,3,5	Set of all prime numbers;
2,3,5,13	Set of all prime numbers;
2,3,5,13,1	Set of all Fibonacci numbers;
2,3,5,13,1,6	Set of all Fibonacci numbers;
...	...

Successful: Algorithmic learner outputs a sequence of hypotheses which converges to one correct forever.

Motivation & Numerical Example

- A motivation: Program of Khoussainov & Nerode 1995 re $\approx$ effect of replacing TMs by finite automata in computable model theory.
- Present paper: One in a series (Jain, Luo and Stephan 2010; Jain, Ong, Pu, Stephan 2010; Case, Jain, Le, Ong, Semukhin, Stephan LATA-2011) devoted to effect on computability-theoretic learning theory under above replacement.
- Example unrestricted learning from positive data:

Data	Hypotheses
2	Set of all even numbers;
2,3	Set of all numbers;
2,3,5	Set of all prime numbers;
2,3,5,13	Set of all prime numbers;
2,3,5,13,1	Set of all finite sets;
2,3,5,13,1,6	Set of all finite sets;
...	...

Successful Algorithmic learners: outputs a sequence of hypotheses which converges to a correct hypothesis.

Motivation & Numerical Example

- A motivation: Program of Khoussainov & Nerode 1995 re $\approx$ effect of replacing TMs by finite automata in computable model theory.
- Present paper: One in a series (Jain, Luo and Stephan 2010; Jain, Ong, Pu, Stephan 2010; Case, Jain, Le, Ong, Semukhin, Stephan LATA-2011) devoted to effect on computability-theoretic learning theory under above replacement.
- Example unrestricted learning from positive data:

Data	Hypotheses
2	Set of all even numbers;
2,3	Set of all numbers;
2,3,5	Set of all prime numbers;
2,3,5,13	Set of all prime numbers;
2,3,5,13,1	Set of all Fibonacci numbers;
2,3,5,13,1,8	Set of all Fibonacci numbers.
...	...

Success: Algorithmic learner outputs a sequence of hypotheses which eventually stabilizes on a correct hypothesis.

Motivation & Numerical Example

- A motivation: Program of Khoussainov & Nerode 1995 re $\approx$ effect of replacing TMs by finite automata in computable model theory.
- Present paper: One in a series (Jain, Luo and Stephan 2010; Jain, Ong, Pu, Stephan 2010; Case, Jain, Le, Ong, Semukhin, Stephan LATA-2011) devoted to effect on computability-theoretic learning theory under above replacement.
- Example unrestricted learning from positive data:

Data	Hypotheses
2	Set of all even numbers;
2,3	Set of all numbers;
2,3,5	Set of all prime numbers;
2,3,5,13	Set of all prime numbers;
2,3,5,13,1	Set of all Fibonacci numbers;
2,3,5,13,1,8	Set of all Fibonacci numbers.
...	...

Success: Algorithmic learner outputs a sequence of hypotheses which eventually stabilizes on a correct hypothesis.

Motivation & Numerical Example

- A motivation: Program of Khoussainov & Nerode 1995 re $\approx$ effect of replacing TMs by finite automata in computable model theory.
- Present paper: One in a series (Jain, Luo and Stephan 2010; Jain, Ong, Pu, Stephan 2010; Case, Jain, Le, Ong, Semukhin, Stephan LATA-2011) devoted to effect on computability-theoretic learning theory under above replacement.
- Example unrestricted learning from positive data:

Data	Hypotheses
2	Set of all even numbers;
2,3	Set of all numbers;
2,3,5	Set of all prime numbers;
2,3,5,13	Set of all prime numbers;
2,3,5,13,1	Set of all Fibonacci numbers;
2,3,5,13,1,8	Set of all Fibonacci numbers.
...	...

Success: Algorithmic learner outputs a sequence of hypotheses which eventually stabilizes on a correct hypothesis.

Motivation & Numerical Example

- A motivation: Program of Khoussainov & Nerode 1995 re $\approx$ effect of replacing TMs by finite automata in computable model theory.
- Present paper: One in a series (Jain, Luo and Stephan 2010; Jain, Ong, Pu, Stephan 2010; Case, Jain, Le, Ong, Semukhin, Stephan LATA-2011) devoted to effect on computability-theoretic learning theory under above replacement.
- Example unrestricted learning from positive data:

Data	Hypotheses
2	Set of all even numbers;
2,3	Set of all numbers;
2,3,5	Set of all prime numbers;
2,3,5,13	Set of all prime numbers;
2,3,5,13,1	Set of all Fibonacci numbers;
2,3,5,13,1,8	Set of all Fibonacci numbers.
...	...

Success: Algorithmic learner outputs a sequence of hypotheses which eventually stabilizes on a correct hypothesis.

Motivation & Numerical Example

- A motivation: Program of Khoussainov & Nerode 1995 re $\approx$ effect of replacing TMs by finite automata in computable model theory.
- Present paper: One in a series (Jain, Luo and Stephan 2010; Jain, Ong, Pu, Stephan 2010; Case, Jain, Le, Ong, Semukhin, Stephan LATA-2011) devoted to effect on computability-theoretic learning theory under above replacement.
- Example unrestricted learning from positive data:

Data	Hypotheses
2	Set of all even numbers;
2,3	Set of all numbers;
2,3,5	Set of all prime numbers;
2,3,5,13	Set of all prime numbers;
2,3,5,13,1	Set of all Fibonacci numbers;
2,3,5,13,1,8	Set of all Fibonacci numbers.
...	...

Success: Algorithmic learner outputs a sequence of hypotheses which eventually stabilizes on a correct hypothesis.

Motivation & Numerical Example

- A motivation: Program of Khoussainov & Nerode 1995 re $\approx$ effect of replacing TMs by finite automata in computable model theory.
- Present paper: One in a series (Jain, Luo and Stephan 2010; Jain, Ong, Pu, Stephan 2010; Case, Jain, Le, Ong, Semukhin, Stephan LATA-2011) devoted to effect on computability-theoretic learning theory under above replacement.
- Example unrestricted learning from positive data:

Data	Hypotheses
2	Set of all even numbers;
2,3	Set of all numbers;
2,3,5	Set of all prime numbers;
2,3,5,13	Set of all prime numbers;
2,3,5,13,1	Set of all Fibonacci numbers;
2,3,5,13,1,8	Set of all Fibonacci numbers.
...	...

Success: Algorithmic learner outputs a sequence of hypotheses which eventually stabilizes on a correct hypothesis.

Motivation & Numerical Example

- A motivation: Program of Khoussainov & Nerode 1995 re $\approx$ effect of replacing TMs by finite automata in computable model theory.
- Present paper: One in a series (Jain, Luo and Stephan 2010; Jain, Ong, Pu, Stephan 2010; Case, Jain, Le, Ong, Semukhin, Stephan LATA-2011) devoted to effect on computability-theoretic learning theory under above replacement.
- Example unrestricted learning from positive data:

Data	Hypotheses
2	Set of all even numbers;
2,3	Set of all numbers;
2,3,5	Set of all prime numbers;
2,3,5,13	Set of all prime numbers;
2,3,5,13,1	Set of all Fibonacci numbers;
2,3,5,13,1,8	Set of all Fibonacci numbers.
...	...

Success: Algorithmic learner outputs a sequence of hypotheses which eventually stabilizes on a correct hypothesis.

Motivation & Numerical Example

- A motivation: Program of Khoussainov & Nerode 1995 re $\approx$ effect of replacing TMs by finite automata in computable model theory.
- Present paper: One in a series (Jain, Luo and Stephan 2010; Jain, Ong, Pu, Stephan 2010; Case, Jain, Le, Ong, Semukhin, Stephan LATA-2011) devoted to effect on computability-theoretic learning theory under above replacement.
- Example unrestricted learning from positive data:

Data	Hypotheses
2	Set of all even numbers;
2,3	Set of all numbers;
2,3,5	Set of all prime numbers;
2,3,5,13	Set of all prime numbers;
2,3,5,13,1	Set of all Fibonacci numbers;
2,3,5,13,1,8	Set of all Fibonacci numbers.
...	...

Success: Algorithmic learner outputs a sequence of hypotheses which eventually stabilizes on a correct hypothesis.

Semi Computability-Theoretic Setting

- Classes $\mathcal{L}$ of sets L to be learned: (for now) the sets are regular (i.e., accepted by some finite automaton) & subsets of, e.g., $\Sigma^* = \{a, b\}^*$.
- A text T for L is a sequence of all and only the elements of L (plus pause symbol $\# \notin \Sigma$). $\{T(0), T(1), \ldots\} - \{\#\} = L$.
- Learner employs hypotheses $\mathrm{hyp}_t \in J$, J an hypothesis space for (at least) $\mathcal{L}$, and a sequence of long term memories mem_t (each $\in \Gamma^*$).
- Learner has initial long term memory mem_0 and initial hypothesis hyp_0.
- Think of each $t = 0, 1, \ldots$ as a time/stage. Then learner $M : (\mathrm{mem}_t, T(t)) \mapsto (\mathrm{mem}_{t+1}, \mathrm{hyp}_{t+1})$. N.B. TM M could remember the mem_ts, but later in talk M will be a finite automaton — which doesn't remember much.
- Again, M succeeds on L: $(\forall T \text{ for } L)(\exists t)(\forall t' > t)[\mathrm{hyp}_{t'} = \mathrm{hyp}_t \ \&\ \mathrm{hyp}_t \text{ is correct for } L]$.
- With unrestricted memory and M a TM, the just above success criterion is equivalent to the main one from (Gold1967).

Semi Computability-Theoretic Setting

- Classes $\mathcal{L}$ of sets L to be learned: (for now) the sets are regular (i.e., accepted by some finite automaton) & subsets of, e.g., $\Sigma^* = \{a, b\}^*$.
- A text T for L is a sequence of all and only the elements of L (plus pause symbol $\# \notin \Sigma$). $\{T(0), T(1), \ldots\} - \{\#\} = L$.
- Learner employs hypotheses $\mathrm{hyp}_t \in J$, J an hypothesis space for (at least) $\mathcal{L}$, and a sequence of long term memories mem_t (each $\in \Gamma^*$).
- Learner has initial long term memory mem_0 and initial hypothesis hyp_0.
- Think of each $t = 0, 1, \ldots$ as a time/stage. Then learner $M : (\mathrm{mem}_t, T(t)) \mapsto (\mathrm{mem}_{t+1}, \mathrm{hyp}_{t+1})$. N.B. TM M could remember the mem_ts, but later in talk M will be a finite automaton — which doesn't remember much.
- Again, M succeeds on L: $(\forall T \text{ for } L)(\exists t)(\forall t' > t)[\mathrm{hyp}_{t'} = \mathrm{hyp}_t \ \&\ \mathrm{hyp}_t \text{ is correct for } L]$.
- With unrestricted memory and M a TM, the just above success criterion is equivalent to the main one from (Gold1967).

Semi Computability-Theoretic Setting

- Classes $\mathcal{L}$ of sets L to be learned: (for now) the sets are regular (i.e., accepted by some finite automaton) & subsets of, e.g., $\Sigma^* = \{a, b\}^*$.
- A text T for L is a sequence of all and only the elements of L (plus pause symbol $\# \notin \Sigma$). $\{T(0), T(1), \ldots\} - \{\#\} = L$.
- Learner employs hypotheses $\mathrm{hyp}_t \in J$, J an hypothesis space for (at least) $\mathcal{L}$, and a sequence of long term memories mem_t (each $\in \Gamma^*$).
- Learner has initial long term memory mem_0 and initial hypothesis hyp_0.
- Think of each $t = 0, 1, \ldots$ as a time/stage. Then learner $M : (\mathrm{mem}_t, T(t)) \mapsto (\mathrm{mem}_{t+1}, \mathrm{hyp}_{t+1})$. N.B. TM M could remember the mem_ts, but later in talk M will be a finite automaton — which doesn't remember much.
- Again, M succeeds on L: $(\forall T \text{ for } L)(\exists t)(\forall t' > t)[\mathrm{hyp}_{t'} = \mathrm{hyp}_t \ \&\ \mathrm{hyp}_t \text{ is correct for } L]$.
- With unrestricted memory and M a TM, the just above success criterion is equivalent to the main one from (Gold1967).

Semi Computability-Theoretic Setting

- Classes $\mathcal{L}$ of sets L to be learned: (for now) the sets are regular (i.e., accepted by some finite automaton) & subsets of, e.g., $\Sigma^* = \{a, b\}^*$.
- A text T for L is a sequence of all and only the elements of L (plus pause symbol $\# \notin \Sigma$). $\{T(0), T(1), \ldots\} - \{\#\} = L$.
- Learner employs hypotheses $\mathrm{hyp}_t \in J$, J an hypothesis space for (at least) $\mathcal{L}$, and a sequence of long term memories mem_t (each $\in \Gamma^*$).
- Learner has initial long term memory mem_0 and initial hypothesis hyp_0.
- Think of each $t = 0, 1, \ldots$ as a time/stage. Then learner $M : (\mathrm{mem}_t, T(t)) \mapsto (\mathrm{mem}_{t+1}, \mathrm{hyp}_{t+1})$. N.B. TM M could remember the mem_ts, but later in talk M will be a finite automaton — which doesn't remember much.
- Again, M succeeds on L:
  $(\forall T \text{ for } L)(\exists t)(\forall t' > t)[\mathrm{hyp}_{t'} = \mathrm{hyp}_t \ \ \& \ \ \mathrm{hyp}_t \text{ is correct for } L]$.
- With unrestricted memory and M a TM, the just above success criterion is equivalent to the main one from (Gold, 1967).

Semi Computability-Theoretic Setting

- Classes $\mathcal{L}$ of sets L to be learned: (for now) the sets are regular (i.e., accepted by some finite automaton) & subsets of, e.g., $\Sigma^* = \{a, b\}^*$.
- A text T for L is a sequence of all and only the elements of L (plus pause symbol $\# \notin \Sigma$). $\{T(0), T(1), \ldots\} - \{\#\} = L$.
- Learner employs hypotheses $\mathrm{hyp}_t \in J$, J an hypothesis space for (at least) $\mathcal{L}$, and a sequence of long term memories mem_t (each $\in \Gamma^*$).
- Learner has initial long term memory mem_0 and initial hypothesis hyp_0.
- Think of each $t = 0, 1, \ldots$ as a time/stage. Then learner $M : (\mathrm{mem}_t, T(t)) \mapsto (\mathrm{mem}_{t+1}, \mathrm{hyp}_{t+1})$. N.B. TM M could remember the mem_ts, but later in talk M will be a finite automaton — which doesn't remember much.
- Again, M succeeds on L:
  $(\forall T \text{ for } L)(\exists t)(\forall t' > t)[\mathrm{hyp}_{t'} = \mathrm{hyp}_t \ \&\ \mathrm{hyp}_t \text{ is correct for } L]$.
- With unrestricted memory and M a TM, the just above success criterion is equivalent to the main one from (Gold, 1967).

Semi Computability-Theoretic Setting

- Classes $\mathcal{L}$ of sets L to be learned: (for now) the sets are regular (i.e., accepted by some finite automaton) & subsets of, e.g., $\Sigma^* = \{a, b\}^*$.
- A text T for L is a sequence of all and only the elements of L (plus pause symbol $\# \notin \Sigma$). $\{T(0), T(1), \ldots\} - \{\#\} = L$.
- Learner employs hypotheses $\mathrm{hyp}_t \in J$, J an hypothesis space for (at least) $\mathcal{L}$, and a sequence of long term memories mem_t (each $\in \Gamma^*$).
- Learner has initial long term memory mem_0 and initial hypothesis hyp_0.
- Think of each $t = 0, 1, \ldots$ as a time/stage. Then learner $M : (\mathrm{mem}_t, T(t)) \mapsto (\mathrm{mem}_{t+1}, \mathrm{hyp}_{t+1})$. N.B. TM M could remember the mem_ts, but later in talk M will be a finite automaton — which doesn't remember much.
- Again, M succeeds on L:
  $(\forall T \text{ for } L)(\exists t)(\forall t' > t)[\mathrm{hyp}_{t'} = \mathrm{hyp}_t \ \ \& \ \ \mathrm{hyp}_t \text{ is correct for } L]$.
- With unrestricted memory and M a TM, the just above success criterion is equivalent to the main one from (Gold, 1967).

Semi Computability-Theoretic Setting

- Classes $\mathcal{L}$ of sets L to be learned: (for now) the sets are regular (i.e., accepted by some finite automaton) & subsets of, e.g., $\Sigma^* = \{a, b\}^*$.
- A text T for L is a sequence of all and only the elements of L (plus pause symbol $\# \notin \Sigma$). $\{T(0), T(1), \ldots\} - \{\#\} = L$.
- Learner employs hypotheses $\mathrm{hyp}_t \in J$, J an hypothesis space for (at least) $\mathcal{L}$, and a sequence of long term memories mem_t (each $\in \Gamma^*$).
- Learner has initial long term memory mem_0 and initial hypothesis hyp_0.
- Think of each $t = 0, 1, \ldots$ as a time/stage. Then learner $M : (\mathrm{mem}_t, T(t)) \mapsto (\mathrm{mem}_{t+1}, \mathrm{hyp}_{t+1})$. N.B. TM M could remember the mem_ts, but later in talk M will be a finite automaton — which doesn't remember much.
- Again, M succeeds on L:
  $(\forall T \text{ for } L)(\exists t)(\forall t' > t)[\mathrm{hyp}_{t'} = \mathrm{hyp}_t \ \ \& \ \ \mathrm{hyp}_t \text{ is correct for } L]$.
- With unrestricted memory and M a TM, the just above success criterion is equivalent to the main one from (Gold, 1967).

Learnable Classes of Regular Languages

- The Class of All Finite Subsets of Σ^* is TM-learnable: After having seen $T[n] \stackrel{\text{def}}{=} T(0), T(1), \ldots, T(n-1)$, the learner outputs a (canonical) index for the set $(\{T(0), T(1), \ldots, T(n-1)\} - \{\#\})$ of all the (non-#) data seen so far.
- Any class containing (just above class $\cup\{\Sigma^*\}$) is NOT learnable: by a finite extension (Baire Category) argument; hence, the entire class of all regular languages is NOT learnable (Gold 1967)!
- However, many interesting/useful proper subclasses of the class of all regular languages are TM learnable (Angluin 1980; Head, Kobayashi and Yokomori 1998; Fernau 2003, etc).

Learnable Classes of Regular Languages

- The Class of All Finite Subsets of Σ^* is TM-learnable: After having seen $T[n] \stackrel{\text{def}}{=} T(0), T(1), \ldots, T(n-1)$, the learner outputs a (canonical) index for the set $(\{T(0), T(1), \ldots, T(n-1)\} - \{\#\})$ of all the (non-#) data seen so far.
- Any class containing (just above class $\cup\{\Sigma^*\}$) is NOT learnable: by a finite extension (Baire Category) argument; hence, the entire class of all regular languages is NOT learnable (Gold 1967)!
- However, many interesting/useful proper subclasses of the class of all regular languages are TM learnable (Angluin 1980; Head, Kobayashi and Yokomori 1998; Fernau 2003, etc).

Learnable Classes of Regular Languages

- The Class of All Finite Subsets of Σ^* is TM-learnable: After having seen $T[n] \stackrel{\text{def}}{=} T(0), T(1), \ldots, T(n-1)$, the learner outputs a (canonical) index for the set $(\{T(0), T(1), \ldots, T(n-1)\} - \{\#\})$ of all the (non-#) data seen so far.
- Any class containing (just above class $\cup\{\Sigma^*\}$) is NOT learnable: by a finite extension (Baire Category) argument; hence, the entire class of all regular languages is NOT learnable (Gold 1967)!
- However, many interesting/useful proper subclasses of the class of all regular languages are TM learnable (Angluin 1980; Head, Kobayashi and Yokomori 1998; Fernau 2003, etc).

Automatic Structures

- CONVOLUTION: Given $\alpha, \beta \in \Upsilon^*, \Upsilon \supseteq (\Sigma \cup \Gamma \cup \{\#\})$ & $\sqcup \notin \Upsilon$, $\mathrm{conv}(\alpha, \beta) \stackrel{\mathrm{def}}{=} (\alpha(0), \beta(0))(\alpha(1), \beta(1)) \cdots (\alpha(n)\beta(n))$, where $n = \max\{|\alpha|, |\beta|\} - 1$, and, for $m \leq n$, $\alpha(m) = \sqcup$ for $m \geq |\alpha|$ & $\beta(m) = \sqcup$ for $m \geq |\beta|$.
  Example: $\mathrm{conv}(ab, bbb) = (a, b)(b, b)(\sqcup, b)$. Idea: These pairs are new alphabet symbols to be read one after the other by a finite automaton.
- A binary relation R is automatic iff $\{\mathrm{conv}(\alpha, \beta) : R(\alpha, \beta)\}$ is regular — over the alphabet $((\Upsilon \cup \{\sqcup\}) \times (\Upsilon \cup \{\sqcup\}))$.
  The concept obviously generalizes to k-ary relations.
- A function f is automatic iff the relation $\{\mathrm{conv}(w, f(w)) : w \in \mathrm{Domain}(f)\}$ is regular.
  This models how we "compute" functions by finite automata!
- Automatic Structures:
  The domain is regular and all the relations are automatic.

Automatic Structures

- CONVOLUTION: Given $\alpha, \beta \in \Upsilon^*, \Upsilon \supseteq (\Sigma \cup \Gamma \cup \{\#\})$ & $\sqcup \notin \Upsilon$, $\mathrm{conv}(\alpha, \beta) \stackrel{\mathrm{def}}{=} (\alpha(0), \beta(0))(\alpha(1), \beta(1)) \cdots (\alpha(n)\beta(n))$, where $n = \max\{|\alpha|, |\beta|\} - 1$, and, for $m \leq n$, $\alpha(m) = \sqcup$ for $m \geq |\alpha|$ & $\beta(m) = \sqcup$ for $m \geq |\beta|$.
  Example: $\mathrm{conv}(ab, bbb) = (a, b)(b, b)(\sqcup, b)$. Idea: These pairs are new alphabet symbols to be read one after the other by a finite automaton.
- A binary relation R is automatic iff $\{\mathrm{conv}(\alpha, \beta) : R(\alpha, \beta)\}$ is regular — over the alphabet $((\Upsilon \cup \{\sqcup\}) \times (\Upsilon \cup \{\sqcup\}))$.
  The concept obviously generalizes to k-ary relations.
- A function f is automatic iff the relation $\{\mathrm{conv}(w, f(w)) : w \in \mathrm{Domain}(f)\}$ is regular.
  This models how we "compute" functions by finite automata!
- Automatic Structures:
  The domain is regular and all the relations are automatic.

Automatic Structures

- CONVOLUTION: Given $\alpha, \beta \in \Upsilon^*, \Upsilon \supseteq (\Sigma \cup \Gamma \cup \{\#\})$ & $\sqcup \notin \Upsilon$, $\mathrm{conv}(\alpha, \beta) \stackrel{\mathrm{def}}{=} (\alpha(0), \beta(0))(\alpha(1), \beta(1)) \cdots (\alpha(n)\beta(n))$, where $n = \max\{|\alpha|, |\beta|\} - 1$, and, for $m \leq n$, $\alpha(m) = \sqcup$ for $m \geq |\alpha|$ & $\beta(m) = \sqcup$ for $m \geq |\beta|$.
  Example: $\mathrm{conv}(ab, bbb) = (a, b)(b, b)(\sqcup, b)$. Idea: These pairs are new alphabet symbols to be read one after the other by a finite automaton.
- A binary relation R is automatic iff $\{\mathrm{conv}(\alpha, \beta) : R(\alpha, \beta)\}$ is regular — over the alphabet $((\Upsilon \cup \{\sqcup\}) \times (\Upsilon \cup \{\sqcup\}))$.
  The concept obviously generalizes to k-ary relations.
- A function f is automatic iff the relation $\{\mathrm{conv}(w, f(w)) : w \in \mathrm{Domain}(f)\}$ is regular.
  This models how we "compute" functions by finite automata!
- Automatic Structures:
  The domain is regular and all the relations are automatic.

Automatic Structures

- CONVOLUTION: Given $\alpha, \beta \in \Upsilon^*, \Upsilon \supseteq (\Sigma \cup \Gamma \cup \{\#\})$ & $\sqcup \notin \Upsilon$, $\mathrm{conv}(\alpha, \beta) \stackrel{\mathrm{def}}{=} (\alpha(0), \beta(0))(\alpha(1), \beta(1)) \cdots (\alpha(n)\beta(n))$, where $n = \max\{|\alpha|, |\beta|\} - 1$, and, for $m \leq n$, $\alpha(m) = \sqcup$ for $m \geq |\alpha|$ & $\beta(m) = \sqcup$ for $m \geq |\beta|$.
  Example: $\mathrm{conv}(ab, bbb) = (a, b)(b, b)(\sqcup, b)$. Idea: These pairs are new alphabet symbols to be read one after the other by a finite automaton.
- A binary relation R is automatic iff $\{\mathrm{conv}(\alpha, \beta) : R(\alpha, \beta)\}$ is regular — over the alphabet $((\Upsilon \cup \{\sqcup\}) \times (\Upsilon \cup \{\sqcup\}))$.
  The concept obviously generalizes to k-ary relations.
- A function f is automatic iff the relation $\{\mathrm{conv}(w, f(w)) : w \in \mathrm{Domain}(f)\}$ is regular.
  This models how we "compute" functions by finite automata!
- Automatic Structures:
  The domain is regular and all the relations are automatic.

Automatic Classes

$\mathcal{L}$ is an automatic class iff each $L \in \mathcal{L}$ is a subset of Σ^*, and, for some regular index domain I and some regular $S \subseteq I \times \Sigma^*$, $\mathcal{L} = \{L_\alpha : \alpha \in I\}$, where $L_\alpha = \{x : \mathrm{conv}(\alpha, x) \in S\}$.

Idea: Such a $\mathcal{L}$ is uniformly regular.

Examples:

- The class of all sets $x\Sigma^*$ is automatic where the string x could be used as the index.
- The class of all sets $\{z \in \Sigma^* : x \leq_{lex} z \leq_{lex} y\}$ is automatic where the convolution of x and y can be used as an index for the corresponding interval.
- The class of all finite subsets of $\{2,3\}^*$ is NOT automatic; however, the class of all finite subsets of $\{2\}^*$ is automatic (with the indices ranging over $I = (0^*1)^*$ and, where: for any n, $2^n \in L_\alpha$ iff $n < |\alpha|$ & the symbol in α at position n is 1).

From now on: We care about somehow learning automatic classes.

Automatic Classes

$\mathcal{L}$ is an automatic class iff each $L \in \mathcal{L}$ is a subset of Σ^*, and, for some regular index domain I and some regular $S \subseteq I \times \Sigma^*$,
$\mathcal{L} = \{L_\alpha : \alpha \in I\}$, where $L_\alpha = \{x : \mathrm{conv}(\alpha, x) \in S\}$.
Idea: Such a $\mathcal{L}$ is uniformly regular.
Examples:

- The class of all sets $x\Sigma^*$ is automatic where the string x could be used as the index.
- The class of all sets $\{z \in \Sigma^* : x \leq_{lex} z \leq_{lex} y\}$ is automatic where the convolution of x and y can be used as an index for the corresponding interval.
- The class of all finite subsets of $\{2,3\}^*$ is NOT automatic; however, the class of all finite subsets of $\{2\}^*$ is automatic (with the indices ranging over $I = (0^*1)^*$ and, where: for any n, $2^n \in L_\alpha$ iff $n < |\alpha|$ & the symbol in α at position n is 1).

From now on: We care about somehow learning automatic classes.

Automatic Classes

$\mathcal{L}$ is an automatic class iff each $L \in \mathcal{L}$ is a subset of Σ^*, and, for some regular index domain I and some regular $S \subseteq I \times \Sigma^*$,
$\mathcal{L} = \{L_\alpha : \alpha \in I\}$, where $L_\alpha = \{x : \mathrm{conv}(\alpha, x) \in S\}$.
Idea: Such a $\mathcal{L}$ is uniformly regular.
Examples:

- The class of all sets $x\Sigma^*$ is automatic where the string x could be used as the index.
- The class of all sets $\{z \in \Sigma^* : x \leq_{lex} z \leq_{lex} y\}$ is automatic where the convolution of x and y can be used as an index for the corresponding interval.
- The class of all finite subsets of $\{2,3\}^*$ is NOT automatic; however, the class of all finite subsets of $\{2\}^*$ is automatic (with the indices ranging over $I = (0^*1)^*$ and, where: for any n, $2^n \in L_\alpha$ iff $n < |\alpha|$ & the symbol in α at position n is 1).

From now on: We care about somehow learning automatic classes.

Automatic Classes

$\mathcal{L}$ is an automatic class iff each $L \in \mathcal{L}$ is a subset of Σ^*, and, for some regular index domain I and some regular $S \subseteq I \times \Sigma^*$,
$\mathcal{L} = \{L_\alpha : \alpha \in I\}$, where $L_\alpha = \{x : \mathrm{conv}(\alpha, x) \in S\}$.
Idea: Such a $\mathcal{L}$ is uniformly regular.
Examples:

- The class of all sets $x\Sigma^*$ is automatic where the string x could be used as the index.
- The class of all sets $\{z \in \Sigma^* : x \leq_{lex} z \leq_{lex} y\}$ is automatic where the convolution of x and y can be used as an index for the corresponding interval.
- The class of all finite subsets of $\{2,3\}^*$ is NOT automatic; however, the class of all finite subsets of $\{2\}^*$ is automatic (with the indices ranging over $I = (0^*1)^*$ and, where: for any n, $2^n \in L_\alpha$ iff $n < |\alpha|$ & the symbol in α at position n is 1).

From now on: We care about somehow learning automatic classes.

Automatic Classes

$\mathcal{L}$ is an automatic class iff each $L \in \mathcal{L}$ is a subset of Σ^*, and, for some regular index domain I and some regular $S \subseteq I \times \Sigma^*$, $\mathcal{L} = \{L_\alpha : \alpha \in I\}$, where $L_\alpha = \{x : \mathrm{conv}(\alpha, x) \in S\}$.
Idea: Such a $\mathcal{L}$ is uniformly regular.
Examples:

- The class of all sets $x\Sigma^*$ is automatic where the string x could be used as the index.
- The class of all sets $\{z \in \Sigma^* : x \leq_{lex} z \leq_{lex} y\}$ is automatic where the convolution of x and y can be used as an index for the corresponding interval.
- The class of all finite subsets of $\{2,3\}^*$ is NOT automatic; however, the class of all finite subsets of $\{2\}^*$ is automatic (with the indices ranging over $I = (0^*1)^*$ and, where: for any n, $2^n \in L_\alpha$ iff $n < |\alpha|$ & the symbol in α at position n is 1).

From now on: We care about somehow learning automatic classes.

Learning Automatic Classes & Further Motivation

Theorem (Angluin 1980 Adapted to Automatic Classes)

An automatic class $\{L_\alpha : \alpha \in I\}$ is learnable by a TM iff, for every $\alpha \in I$, there is a finite set $D_\alpha \subseteq L_\alpha$ such that,
for all $\beta \in I$, $\neg[D_\alpha \subseteq L_\beta \subset L_\alpha]$.

- The finite set D_α just above is called a tell-tale for L_α.
- From now on the learners $M : (\mathrm{mem}_n, T(n)) \mapsto (\mathrm{mem}_{n+1}, \mathrm{hyp}_{n+1})$ we focus on will be automatic — w/ hyp_ns in a regular index set for an automatic class.

Proposition (Another Motivation for This Paper)

The Output of an automatic learning function can be TM calculated from its input uniformly in linear time — some cases can be practical.

Learning Automatic Classes & Further Motivation

Theorem (Angluin 1980 Adapted to Automatic Classes)

An automatic class $\{L_\alpha : \alpha \in I\}$ is learnable by a TM iff, for every $\alpha \in I$, there is a finite set $D_\alpha \subseteq L_\alpha$ such that,
for all $\beta \in I$, $\neg[D_\alpha \subseteq L_\beta \subset L_\alpha]$.

- The finite set D_α just above is called a tell-tale for L_α.
- From now on the learners $M : (\mathrm{mem}_n, T(n)) \mapsto (\mathrm{mem}_{n+1}, \mathrm{hyp}_{n+1})$ we focus on will be automatic — w/ hyp_ns in a regular index set for an automatic class.

Proposition (Another Motivation for This Paper)

The Output of an automatic learning function can be TM calculated from its input uniformly in linear time — some cases can be practical.

Learning Automatic Classes & Further Motivation

Theorem (Angluin 1980 Adapted to Automatic Classes)

An automatic class $\{L_\alpha : \alpha \in I\}$ is learnable by a TM iff, for every $\alpha \in I$, there is a finite set $D_\alpha \subseteq L_\alpha$ such that,
for all $\beta \in I$, $\neg[D_\alpha \subseteq L_\beta \subset L_\alpha]$.

- The finite set D_α just above is called a tell-tale for L_α.
- From now on the learners $M : (\mathrm{mem}_n, T(n)) \mapsto (\mathrm{mem}_{n+1}, \mathrm{hyp}_{n+1})$ we focus on will be automatic — w/ hyp_ns in a regular index set for an automatic class.

Proposition (Another Motivation for This Paper)

The Output of an automatic learning function can be TM calculated from its input uniformly in linear time — some cases can be practical.

Memory Restrictions

- Iterative (Wiehagen 1976): Long term memory is the previous hypothesis: $mem_t = \mathrm{hyp}_t$.
- c-Bounded Example-Memory (Osherson, Stob and Weinstein 1986): Long term memory consists of up to c selected input data.
- Example-Bounded (Jain, Luo and Stephan 2010): mem_t is a string of length bounded by the length of the longest example datum seen so far — plus a constant.
- Hypothesis-Bounded (Jain, Luo and Stephan 2010): mem_t is a string of length bounded by the length of hyp_t — plus a constant.
- k-Bounded Feedback Queries (Lange, Zeugmann 1996; Case, Jain, Lange, Zeugmann 1999): Allows asking, in each round t, which of k computed data items have been seen previously.

How to formulate this in the context of automatic learners (new to the present paper), next frame.

Memory Restrictions

- Iterative (Wiehagen 1976): Long term memory is the previous hypothesis: $mem_t = \mathrm{hyp}_t$.
- c-Bounded Example-Memory (Osherson, Stob and Weinstein 1986): Long term memory consists of up to c selected input data.
- Example-Bounded (Jain, Luo and Stephan 2010): mem_t is a string of length bounded by the length of the longest example datum seen so far — plus a constant.
- Hypothesis-Bounded (Jain, Luo and Stephan 2010): mem_t is a string of length bounded by the length of hyp_t — plus a constant.
- k-Bounded Feedback Queries (Lange, Zeugmann 1996; Case, Jain, Lange, Zeugmann 1999): Allows asking, in each round t, which of k computed data items have been seen previously.
  How to formulate this in the context of automatic learners (new to the present paper), next frame.

Memory Restrictions

- Iterative (Wiehagen 1976): Long term memory is the previous hypothesis: $mem_t = \mathrm{hyp}_t$.
- c-Bounded Example-Memory (Osherson, Stob and Weinstein 1986): Long term memory consists of up to c selected input data.
- Example-Bounded (Jain, Luo and Stephan 2010): mem_t is a string of length bounded by the length of the longest example datum seen so far — plus a constant.
- Hypothesis-Bounded (Jain, Luo and Stephan 2010): mem_t is a string of length bounded by the length of hyp_t — plus a constant.
- k-Bounded Feedback Queries (Lange, Zeugmann 1996; Case, Jain, Lange, Zeugmann 1999): Allows asking, in each round t, which of k computed data items have been seen previously.
  How to formulate this in the context of automatic learners (new to the present paper), next frame.

Memory Restrictions

- Iterative (Wiehagen 1976): Long term memory is the previous hypothesis: $mem_t = \mathrm{hyp}_t$.
- c-Bounded Example-Memory (Osherson, Stob and Weinstein 1986): Long term memory consists of up to c selected input data.
- Example-Bounded (Jain, Luo and Stephan 2010): mem_t is a string of length bounded by the length of the longest example datum seen so far — plus a constant.
- Hypothesis-Bounded (Jain, Luo and Stephan 2010): mem_t is a string of length bounded by the length of hyp_t — plus a constant.
- k-Bounded Feedback Queries (Lange, Zeugmann 1996; Case, Jain, Lange, Zeugmann 1999): Allows asking, in each round t, which of k computed data items have been seen previously.
  How to formulate this in the context of automatic learners (new to the present paper), next frame.

Memory Restrictions

- Iterative (Wiehagen 1976): Long term memory is the previous hypothesis: $mem_t = \mathrm{hyp}_t$.
- c-Bounded Example-Memory (Osherson, Stob and Weinstein 1986): Long term memory consists of up to c selected input data.
- Example-Bounded (Jain, Luo and Stephan 2010): mem_t is a string of length bounded by the length of the longest example datum seen so far — plus a constant.
- Hypothesis-Bounded (Jain, Luo and Stephan 2010): mem_t is a string of length bounded by the length of hyp_t — plus a constant.
- k-Bounded Feedback Queries (Lange, Zeugmann 1996; Case, Jain, Lange, Zeugmann 1999): Allows asking, in each round t, which of k computed data items have been seen previously.
  How to formulate this in the context of automatic learners (new to the present paper), next frame.

Formulate Automatic Feedback Learning

Re Automatic k-Bounded Feedback Query Learning: We do NOT employ $\mathrm{mem}_n = \mathrm{conv}(T[n])$ since that would require bigger memory alphabets for bigger n. Instead:

For Automatic k-Bounded Feedback Query Learning one has:

- An automatic query function $Q : (\mathrm{mem}_n, T(n)) \mapsto (q_1, \ldots, q_k)$, where each q_i lies in the underlying regular domain. Q decides what queries to ask.
- For $1 \leq i \leq k$, bit $b_i = 1$ iff $q_i \in \{T(0), T(1), \ldots, T(n-1)\}$.
- A special, associated automatic learner M: $(\mathrm{mem}_n, T(n), b_1, \ldots, b_k) \mapsto (\mathrm{mem}_{n+1}, \mathrm{hyp}_{n+1})$.

Formulate Automatic Feedback Learning

Re Automatic k-Bounded Feedback Query Learning: We do NOT employ $\mathrm{mem}_n = \mathrm{conv}(T[n])$ since that would require bigger memory alphabets for bigger n. Instead:
For Automatic k-Bounded Feedback Query Learning one has:

- An automatic query function $Q : (\mathrm{mem}_n, T(n)) \mapsto (q_1, \ldots, q_k)$, where each q_i lies in the underlying regular domain. Q decides what queries to ask.
- For $1 \leq i \leq k$, bit $b_i = 1$ iff $q_i \in \{T(0), T(1), \ldots, T(n-1)\}$.
- A special, associated automatic learner M: $(\mathrm{mem}_n, T(n), b_1, \ldots, b_k) \mapsto (\mathrm{mem}_{n+1}, \mathrm{hyp}_{n+1})$.

Formulate Automatic Feedback Learning

Re Automatic k-Bounded Feedback Query Learning: We do NOT employ $\mathrm{mem}_n = \mathrm{conv}(T[n])$ since that would require bigger memory alphabets for bigger n. Instead:
For Automatic k-Bounded Feedback Query Learning one has:

- An automatic query function $Q : (\mathrm{mem}_n, T(n)) \mapsto (q_1, \ldots, q_k)$, where each q_i lies in the underlying regular domain. Q decides what queries to ask.
- For $1 \leq i \leq k$, bit $b_i = 1$ iff $q_i \in \{T(0), T(1), \ldots, T(n-1)\}$.
- A special, associated automatic learner M: $(\mathrm{mem}_n, T(n), b_1, \ldots, b_k) \mapsto (\mathrm{mem}_{n+1}, \mathrm{hyp}_{n+1})$.

Formulate Automatic Feedback Learning

Re Automatic k-Bounded Feedback Query Learning: We do NOT employ $\mathrm{mem}_n = \mathrm{conv}(T[n])$ since that would require bigger memory alphabets for bigger n. Instead:
For Automatic k-Bounded Feedback Query Learning one has:

- An automatic query function $Q : (\mathrm{mem}_n, T(n)) \mapsto (q_1, \ldots, q_k)$, where each q_i lies in the underlying regular domain. Q decides what queries to ask.
- For $1 \leq i \leq k$, bit $b_i = 1$ iff $q_i \in \{T(0), T(1), \ldots, T(n-1)\}$.
- A special, associated automatic learner M: $(\mathrm{mem}_n, T(n), b_1, \ldots, b_k) \mapsto (\mathrm{mem}_{n+1}, \mathrm{hyp}_{n+1})$.

Examples

- While $COSINGLE = \{\{0,1\}^* - \{x\} : x \in \{0,1\}^*\}$, is automatic, it does NOT have a normal automatic learner (Jain, Luo, Stephan 2010).
  However, it can be learned by an automatic feedback learner (using one query per round): learner converges to hypothesis for $\{0,1\}^* - \{x\}$, for the length-lexicographically least member x of $\{0,1\}^*$ for which the feedback query answer remains negative forever. Successive candidates for such x are stored in the mem_ns.
- The family of the closed intervals $L_{conv(x,y)} = \{z \in \Sigma^* : x \leq_{lex} z \leq_{lex} y\}$ is noted above to be automatic. It can be learned by an automatic learner with 2-bounded example-memory.
- The family of the open intervals $L_{conv(x,y)} = \{z \in \Sigma^* : x <_{lex} z <_{lex} y\}$ is also automatic. However, it canNOT be learned period since it violates Angluin's tell-tale condition.

Examples

- While $COSINGLE = \{\{0,1\}^* - \{x\} : x \in \{0,1\}^*\}$, is automatic, it does NOT have a normal automatic learner (Jain, Luo, Stephan 2010).
  However, it can be learned by an automatic feedback learner (using one query per round): learner converges to hypothesis for $\{0,1\}^* - \{x\}$, for the length-lexicographically least member x of $\{0,1\}^*$ for which the feedback query answer remains negative forever. Successive candidates for such x are stored in the mem_ns.
- The family of the closed intervals $L_{conv(x,y)} = \{z \in \Sigma^* : x \leq_{lex} z \leq_{lex} y\}$ is noted above to be automatic. It can be learned by an automatic learner with 2-bounded example-memory.
- The family of the open intervals $L_{conv(x,y)} = \{z \in \Sigma^* : x <_{lex} z <_{lex} y\}$ is also automatic. However, it canNOT be learned period since it violates Angluin's tell-tale condition.

Examples

- While $COSINGLE = \{\{0,1\}^* - \{x\} : x \in \{0,1\}^*\}$, is automatic, it does NOT have a normal automatic learner (Jain, Luo, Stephan 2010).
  However, it can be learned by an automatic feedback learner (using one query per round): learner converges to hypothesis for $\{0,1\}^* - \{x\}$, for the length-lexicographically least member x of $\{0,1\}^*$ for which the feedback query answer remains negative forever. Successive candidates for such x are stored in the mem_ns.
- The family of the closed intervals $L_{conv(x,y)} = \{z \in \Sigma^* : x \leq_{lex} z \leq_{lex} y\}$ is noted above to be automatic. It can be learned by an automatic learner with 2-bounded example-memory.
- The family of the open intervals $L_{conv(x,y)} = \{z \in \Sigma^* : x <_{lex} z <_{lex} y\}$ is also automatic. However, it canNOT be learned period since it violates Angluin's tell-tale condition.

Results

Theorem

If automatic class $\mathcal{L}$ satisfies Angluin's tell-tale condition, then $\mathcal{L}$ can be learned by an automatic learner with 1-feedback query per round and a long term memory bounded by the longest word seen so far — plus a constant.

Hence, by the adapted Angluin result above, automatic classes TM-learnable have an automatic one query feedback learner with liberal memory as just described!

Theorem

There is an automatic class $\mathcal{L}$ satisfying:

- An automatic learner with each $\mathrm{mem}_t = \varepsilon$ and using one-feedback query per round, can learn $\mathcal{L}$; and
- No normal automatic learner, with unrestricted mem_ts, learns $\mathcal{L}$.

Results

Theorem

If automatic class $\mathcal{L}$ satisfies Angluin's tell-tale condition, then $\mathcal{L}$ can be learned by an automatic learner with 1-feedback query per round and a long term memory bounded by the longest word seen so far — plus a constant.
Hence, by the adapted Angluin result above, automatic classes TM-learnable have an automatic one query feedback learner with liberal memory as just described!

Theorem

There is an automatic class $\mathcal{L}$ satisfying:

- An automatic learner with each $\mathrm{mem}_t = \varepsilon$ and using one-feedback query per round, can learn $\mathcal{L}$; and
- No normal automatic learner, with unrestricted mem_ts, learns $\mathcal{L}$.

Results

Theorem

If automatic class $\mathcal{L}$ satisfies Angluin's tell-tale condition, then $\mathcal{L}$ can be learned by an automatic learner with 1-feedback query per round and a long term memory bounded by the longest word seen so far — plus a constant.
Hence, by the adapted Angluin result above, automatic classes TM-learnable have an automatic one query feedback learner with liberal memory as just described!

Theorem

There is an automatic class $\mathcal{L}$ satisfying:

- An automatic learner with each $\mathrm{mem}_t = \varepsilon$ and using one-feedback query per round, can learn $\mathcal{L}$; and
- No normal automatic learner, with unrestricted mem_ts, learns $\mathcal{L}$.

Results

Theorem

If automatic class $\mathcal{L}$ satisfies Angluin's tell-tale condition, then $\mathcal{L}$ can be learned by an automatic learner with 1-feedback query per round and a long term memory bounded by the longest word seen so far — plus a constant.
Hence, by the adapted Angluin result above, automatic classes TM-learnable have an automatic one query feedback learner with liberal memory as just described!

Theorem

There is an automatic class $\mathcal{L}$ satisfying:

- An automatic learner with each $\mathrm{mem}_t = \varepsilon$ and using one-feedback query per round, can learn $\mathcal{L}$; and
- No normal automatic learner, with unrestricted mem_ts, learns $\mathcal{L}$.

Results Continued

Theorem

There is an automatic class $\mathcal{L}$ such that:

- $\mathcal{L}$ can be learned by an automatic learner with one-bounded example-memory; $\mathcal{L}$ can also be learned by an automatic learner using two feedback queries per round, with long term memory bounded by the size of the hypothesis (plus a constant); and
- $\mathcal{L}$ canNOT be learned by an automatic ITERATIVE learner using any number k of feedback queries.

Theorem (Hierarchy)

For each $k \geq 1$, some automatic class $\mathcal{L}$ satisfies:
For each $c \in \{0, 1, \ldots, k\}$, $\mathcal{L}$ can be learned by an automatic learner which uses c-bounded example-memory & $k - c$ feedback queries (and no other memory) — but NOT if one of bounded-example memory & feedback is $k - 1$ & the other is 0.

Results Continued

Theorem

There is an automatic class $\mathcal{L}$ such that:

- $\mathcal{L}$ can be learned by an automatic learner with one-bounded example-memory; $\mathcal{L}$ can also be learned by an automatic learner using two feedback queries per round, with long term memory bounded by the size of the hypothesis (plus a constant); and
- $\mathcal{L}$ canNOT be learned by an automatic ITERATIVE learner using any number k of feedback queries.

Theorem (Hierarchy)

For each $k \geq 1$, some automatic class $\mathcal{L}$ satisfies:
For each $c \in \{0, 1, \ldots, k\}$, $\mathcal{L}$ can be learned by an automatic learner which uses c-bounded example-memory & $k - c$ feedback queries (and no other memory) — but NOT if one of bounded-example memory & feedback is $k - 1$ & the other is 0.

Results Continued

Theorem

There is an automatic class $\mathcal{L}$ such that:

- $\mathcal{L}$ can be learned by an automatic learner with one-bounded example-memory; $\mathcal{L}$ can also be learned by an automatic learner using two feedback queries per round, with long term memory bounded by the size of the hypothesis (plus a constant); and
- $\mathcal{L}$ canNOT be learned by an automatic ITERATIVE learner using any number k of feedback queries.

Theorem (Hierarchy)

For each $k \geq 1$, some automatic class $\mathcal{L}$ satisfies:
For each $c \in \{0, 1, \ldots, k\}$, $\mathcal{L}$ can be learned by an automatic learner which uses c-bounded example-memory & $k - c$ feedback queries (and no other memory) — but NOT if one of bounded-example memory & feedback is $k - 1$ & the other is 0.