Difficulties in Forcing Fairness of Polynomial Time Inductive Inference

John Case, **Timo Kötzing**
University of Delaware

October 4, 2009

Examples for Language Learning

We want to learn correct programmable descriptions for given languages, such as:

$16, 12, 18, 2, 4, 0, 16, \ldots$	"even numbers"
$1, 16, 256, 16, 4, \ldots$	"powers of 2"
$0, 0, 0, 0, 0, \ldots$	"singleton 0"

Examples for Language Learning

We want to learn correct programmable descriptions for given languages, such as:

$16, 12, 18, 2, 4, 0, 16, \ldots$	"even numbers"
$1, 16, 256, 16, 4, \ldots$	"powers of 2"
$0, 0, 0, 0, 0, \ldots$	"singleton 0"

Examples for Language Learning

We want to learn correct programmable descriptions for given languages, such as:

$16, 12, 18, 2, 4, 0, 16, \ldots$	"even numbers"
$1, 16, 256, 16, 4, \ldots$	"powers of 2"
$0, 0, 0, 0, 0, \ldots$	"singleton 0"

Examples for Language Learning

We want to learn correct programmable descriptions for given languages, such as:

$16, 12, 18, 2, 4, 0, 16, \dots$	"even numbers"
$1, 16, 256, 16, 4, \dots$	"powers of 2"
$0, 0, 0, 0, 0, \dots$	"singleton 0"

Examples for Language Learning

We want to learn correct programmable descriptions for given languages, such as:

$16, 12, 18, 2, 4, 0, 16, \ldots$	"even numbers"
$1, 16, 256, 16, 4, \ldots$	"powers of 2"
$0, 0, 0, 0, 0, \ldots$	"singleton 0"

Examples for Language Learning

We want to learn correct programmable descriptions for given languages, such as:

$16, 12, 18, 2, 4, 0, 16, \ldots$	"even numbers"
$1, 16, 256, 16, 4, \ldots$	"powers of 2"
$0, 0, 0, 0, 0, \ldots$	"singleton 0"

Examples for Language Learning

We want to learn correct programmable descriptions for given languages, such as:

$16, 12, 18, 2, 4, 0, 16, \ldots$	"even numbers"
$1, 16, 256, 16, 4, \ldots$	"powers of 2"
$0, 0, 0, 0, 0, \ldots$	"singleton 0"

Examples for Language Learning

We want to learn correct programmable descriptions for given languages, such as:

$16, 12, 18, 2, 4, 0, 16, \ldots$	"even numbers"
$1, 16, 256, 16, 4, \ldots$	"powers of 2"
$0, 0, 0, 0, 0, \ldots$	"singleton 0"

Language Learning in the Limit from Positive Data

- Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of all natural numbers.
- A language is a set $L \subseteq \mathbb{N}$.
- A presentation for L is essentially an (infinite) listing T of all and only the elements of L. Such a T is called a text for L.
- A hypothesis space V is essentially a mapping from natural numbers to languages.
- For a natural number e that is mapped to a language L, we think of e as a program for L; further, e can be used as an hypothesis for L.
- For a natural number e, we write V_e for the language that e is mapped to.

Language Learning in the Limit from Positive Data

- Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of all natural numbers.
- A language is a set $L \subseteq \mathbb{N}$.
- A presentation for L is essentially an (infinite) listing T of all and only the elements of L. Such a T is called a text for L.
- A hypothesis space V is essentially a mapping from natural numbers to languages.
- For a natural number e that is mapped to a language L, we think of e as a program for L; further, e can be used as an hypothesis for L.
- For a natural number e, we write V_e for the language that e is mapped to.

Language Learning in the Limit from Positive Data

- Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of all natural numbers.
- A language is a set $L \subseteq \mathbb{N}$.
- A presentation for L is essentially an (infinite) listing T of all and only the elements of L. Such a T is called a text for L.
- A hypothesis space V is essentially a mapping from natural numbers to languages.
- For a natural number e that is mapped to a language L, we think of e as a program for L; further, e can be used as an hypothesis for L.
- For a natural number e, we write V_e for the language that e is mapped to.

Language Learning in the Limit from Positive Data

- Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of all natural numbers.
- A language is a set $L \subseteq \mathbb{N}$.
- A presentation for L is essentially an (infinite) listing T of all and only the elements of L. Such a T is called a text for L.
- A hypothesis space V is essentially a mapping from natural numbers to languages.
- For a natural number e that is mapped to a language L, we think of e as a program for L; further, e can be used as an hypothesis for L.
- For a natural number e, we write V_e for the language that e is mapped to.

Language Learning in the Limit from Positive Data

- Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of all natural numbers.
- A language is a set $L \subseteq \mathbb{N}$.
- A presentation for L is essentially an (infinite) listing T of all and only the elements of L. Such a T is called a text for L.
- A hypothesis space V is essentially a mapping from natural numbers to languages.
- For a natural number e that is mapped to a language L, we think of e as a program for L; further, e can be used as an hypothesis for L.
- For a natural number e, we write V_e for the language that e is mapped to.

Language Learning in the Limit from Positive Data

- Let $\mathbb{N} = \{0, 1, 2, \dots\}$ be the set of all natural numbers.
- A language is a set $L \subseteq \mathbb{N}$.
- A presentation for L is essentially an (infinite) listing T of all and only the elements of L. Such a T is called a text for L.
- A hypothesis space V is essentially a mapping from natural numbers to languages.
- For a natural number e that is mapped to a language L, we think of e as a program for L; further, e can be used as an hypothesis for L.
- For a natural number e, we write V_e for the language that e is mapped to.

Language Learning in the Limit from Positive Data

- Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of all natural numbers.
- A language is a set $L \subseteq \mathbb{N}$.
- A presentation for L is essentially an (infinite) listing T of all and only the elements of L. Such a T is called a text for L.
- A hypothesis space V is essentially a mapping from natural numbers to languages.
- For a natural number e that is mapped to a language L, we think of e as a program for L; further, e can be used as an hypothesis for L.
- For a natural number e, we write V_e for the language that e is mapped to.

Success: TxtEx-Learning

- Let V be an hypothesis space, L a language, h a learner and T a text (a presentation) for L.
- For all k, we write $T[k]$ for the sequence $T(0), \ldots, T(k-1)$.
- The learning sequence p_T of h on T is given by

$$\forall k : p_T(k) = h(T[k]). \tag{1}$$

- Gold 1967: h TxtEx-learns L wrt V iff, for all texts T for L, there is i such that $p_T(i) = p_T(i+1) = p_T(i+2) = \ldots$ and $p_T(i)$ is a program in V for L.
- A set $\mathcal{L}$ of languages is TxtEx-learnable wrt V iff there exists a computable learner h TxtEx-learning wrt V all languages $L \in \mathcal{L}$.

Success: TxtEx-Learning

- Let V be an hypothesis space, L a language, h a learner and T a text (a presentation) for L.
- For all k, we write $T[k]$ for the sequence $T(0), \ldots, T(k-1)$.
- The learning sequence p_T of h on T is given by

$$\forall k : p_T(k) = h(T[k]). \tag{1}$$

- Gold 1967: h TxtEx-learns L wrt V iff, for all texts T for L, there is i such that $p_T(i) = p_T(i+1) = p_T(i+2) = \ldots$ and $p_T(i)$ is a program in V for L.
- A set $\mathcal{L}$ of languages is TxtEx-learnable wrt V iff there exists a computable learner h TxtEx-learning wrt V all languages $L \in \mathcal{L}$.

Success: TxtEx-Learning

- Let V be an hypothesis space, L a language, h a learner and T a text (a presentation) for L.
- For all k, we write $T[k]$ for the sequence $T(0), \ldots, T(k-1)$.
- The learning sequence p_T of h on T is given by

$$\forall k : p_T(k) = h(T[k]). \tag{1}$$

- Gold 1967: h TxtEx-learns L wrt V iff, for all texts T for L, there is i such that $p_T(i) = p_T(i+1) = p_T(i+2) = \ldots$ and $p_T(i)$ is a program in V for L.
- A set $\mathcal{L}$ of languages is TxtEx-learnable wrt V iff there exists a computable learner h TxtEx-learning wrt V all languages $L \in \mathcal{L}$.

Success: TxtEx-Learning

- Let V be an hypothesis space, L a language, h a learner and T a text (a presentation) for L.
- For all k, we write $T[k]$ for the sequence $T(0), \dots, T(k-1)$.
- The learning sequence p_T of h on T is given by

$$\forall k : p_T(k) = h(T[k]). \tag{1}$$

- Gold 1967: h TxtEx-learns L wrt V iff, for all texts T for L, there is i such that $p_T(i) = p_T(i+1) = p_T(i+2) = \dots$ and $p_T(i)$ is a program in V for L.
- A set $\mathcal{L}$ of languages is TxtEx-learnable wrt V iff there exists a computable learner h TxtEx-learning wrt V all languages $L \in \mathcal{L}$.

Success: TxtEx-Learning

- Let V be an hypothesis space, L a language, h a learner and T a text (a presentation) for L.
- For all k, we write $T[k]$ for the sequence $T(0), \dots, T(k-1)$.
- The learning sequence p_T of h on T is given by

$$\forall k : p_T(k) = h(T[k]). \tag{1}$$

- Gold 1967: h TxtEx-learns L wrt V iff, for all texts T for L, there is i such that $p_T(i) = p_T(i+1) = p_T(i+2) = \dots$ and $p_T(i)$ is a program in V for L.
- A set $\mathcal{L}$ of languages is TxtEx-learnable wrt V iff there exists a computable learner h TxtEx-learning wrt V all languages $L \in \mathcal{L}$.

Success: TxtEx-Learning

- Let V be an hypothesis space, L a language, h a learner and T a text (a presentation) for L.
- For all k, we write $T[k]$ for the sequence $T(0), \ldots, T(k-1)$.
- The learning sequence p_T of h on T is given by

$$\forall k : p_T(k) = h(T[k]). \tag{1}$$

- Gold 1967: h TxtEx-learns L wrt V iff, for all texts T for L, there is i such that $p_T(i) = p_T(i+1) = p_T(i+2) = \ldots$ and $p_T(i)$ is a program in V for L.
- A set $\mathcal{L}$ of languages is TxtEx-learnable wrt V iff there exists a computable learner h TxtEx-learning wrt V all languages $L \in \mathcal{L}$.

Success: TxtEx-Learning

- Let V be an hypothesis space, L a language, h a learner and T a text (a presentation) for L.
- For all k, we write $T[k]$ for the sequence $T(0), \ldots, T(k-1)$.
- The learning sequence p_T of h on T is given by

$$\forall k : p_T(k) = h(T[k]). \tag{1}$$

- Gold 1967: h TxtEx-learns L wrt V iff, for all texts T for L, there is i such that $p_T(i) = p_T(i+1) = p_T(i+2) = \ldots$ and $p_T(i)$ is a program in V for L.
- A set $\mathcal{L}$ of languages is TxtEx-learnable wrt V iff there exists a computable learner h TxtEx-learning wrt V all languages $L \in \mathcal{L}$.

Complexity in TxtEx-learning

- For TxtEx-learning, we sometimes want the computation of h on $T[k]$ $(= T(0), \ldots, T(k-1))$ to take no more than polynomial time (in $k + \sum_{i=0}^{k-1} |T(i)|$).
- Fact (Pitt 1989): Essentially, for every TxtEx-learnable set of languages $\mathcal{L}$, there is such a polynomial time computable learner learning $\mathcal{L}$.
- Why? A polynomial time learner can be obtained from delaying necessary computations to a later time, when sufficient computing time is available (due to having a longer input).
- As a result, polynomial time learning as above introduces no actual efficiency, as unfair delaying tricks can be used.
- Hence, correspondingly, we seek to limit unfair delaying tricks.

Complexity in TxtEx-learning

- For TxtEx-learning, we sometimes want the computation of h on $T[k]$ $(= T(0), \dots, T(k-1))$ to take no more than polynomial time (in $k + \sum_{i=0}^{k-1} |T(i)|$).
- Fact (Pitt 1989): Essentially, for every TxtEx-learnable set of languages $\mathcal{L}$, there is such a polynomial time computable learner learning $\mathcal{L}$.
- Why? A polynomial time learner can be obtained from delaying necessary computations to a later time, when sufficient computing time is available (due to having a longer input).
- As a result, polynomial time learning as above introduces no actual efficiency, as unfair delaying tricks can be used.
- Hence, correspondingly, we seek to limit unfair delaying tricks.

Complexity in TxtEx-learning

- For TxtEx-learning, we sometimes want the computation of h on $T[k]$ $(= T(0), \ldots, T(k-1))$ to take no more than polynomial time (in $k + \sum_{i=0}^{k-1} |T(i)|$).
- Fact (Pitt 1989): Essentially, for every TxtEx-learnable set of languages $\mathcal{L}$, there is such a polynomial time computable learner learning $\mathcal{L}$.
- Why? A polynomial time learner can be obtained from delaying necessary computations to a later time, when sufficient computing time is available (due to having a longer input).
- As a result, polynomial time learning as above introduces no actual efficiency, as unfair delaying tricks can be used.
- Hence, correspondingly, we seek to limit unfair delaying tricks.

Complexity in TxtEx-learning

- For TxtEx-learning, we sometimes want the computation of h on $T[k]$ $(= T(0), \ldots, T(k-1))$ to take no more than polynomial time (in $k + \sum_{i=0}^{k-1} |T(i)|$).
- Fact (Pitt 1989): Essentially, for every TxtEx-learnable set of languages $\mathcal{L}$, there is such a polynomial time computable learner learning $\mathcal{L}$.
- Why? A polynomial time learner can be obtained from delaying necessary computations to a later time, when sufficient computing time is available (due to having a longer input).
- As a result, polynomial time learning as above introduces no actual efficiency, as unfair delaying tricks can be used.
- Hence, correspondingly, we seek to limit unfair delaying tricks.

Complexity in TxtEx-learning

- For TxtEx-learning, we sometimes want the computation of h on $T[k]$ ($= T(0), \ldots, T(k-1)$) to take no more than polynomial time (in $k + \sum_{i=0}^{k-1} |T(i)|$).
- Fact (Pitt 1989): Essentially, for every TxtEx-learnable set of languages $\mathcal{L}$, there is such a polynomial time computable learner learning $\mathcal{L}$.
- Why? A polynomial time learner can be obtained from delaying necessary computations to a later time, when sufficient computing time is available (due to having a longer input).
- As a result, polynomial time learning as above introduces no actual efficiency, as unfair delaying tricks can be used.
- Hence, correspondingly, we seek to limit unfair delaying tricks.

Complexity in TxtEx-learning

- For TxtEx-learning, we sometimes want the computation of h on $T[k]$ $(= T(0), \ldots, T(k-1))$ to take no more than polynomial time (in $k + \sum_{i=0}^{k-1} |T(i)|$).
- Fact (Pitt 1989): Essentially, for every TxtEx-learnable set of languages $\mathcal{L}$, there is such a polynomial time computable learner learning $\mathcal{L}$.
- Why? A polynomial time learner can be obtained from delaying necessary computations to a later time, when sufficient computing time is available (due to having a longer input).
- As a result, polynomial time learning as above introduces no actual efficiency, as unfair delaying tricks can be used.
- Hence, correspondingly, we seek to limit unfair delaying tricks.

Restrictions on TxtEx-Learning

- In Yoshinaka 2009 (a very nice paper) it is claimed that the following three restrictions force fairness of polynomial time restricted learning:
- postdictive completeness a.k.a. consistency (i.e., a learner only outputs hypotheses postdicting all known data);
- conservativeness (i.e., a learner revises its hypothesis only when that hypothesis fails to predict a new, current datum) and
- prudence (i.e., a learner's hypotheses are only for languages it can learn).
- Below, we will talk about 3-of-3 to refer to the combination of postdictive completeness, conservativeness and prudence.
- We will talk about <3 to refer to the combination of any of postdictive completeness, conservativeness and prudence, just not all three.

Restrictions on TxtEx-Learning

- In Yoshinaka 2009 (a very nice paper) it is claimed that the following three restrictions force fairness of polynomial time restricted learning:
- postdictive completeness a.k.a. consistency (i.e., a learner only outputs hypotheses postdicting all known data);
- conservativeness (i.e., a learner revises its hypothesis only when that hypothesis fails to predict a new, current datum) and
- prudence (i.e., a learner's hypotheses are only for languages it can learn).
- Below, we will talk about 3-of-3 to refer to the combination of postdictive completeness, conservativeness and prudence.
- We will talk about <3 to refer to the combination of any of postdictive completeness, conservativeness and prudence, just not all three.

Restrictions on TxtEx-Learning

- In Yoshinaka 2009 (a very nice paper) it is claimed that the following three restrictions force fairness of polynomial time restricted learning:
- postdictive completeness a.k.a. consistency (i.e., a learner only outputs hypotheses postdicting all known data);
- conservativeness (i.e., a learner revises its hypothesis only when that hypothesis fails to predict a new, current datum) and
- prudence (i.e., a learner's hypotheses are only for languages it can learn).
- Below, we will talk about 3-of-3 to refer to the combination of postdictive completeness, conservativeness and prudence.
- We will talk about <3 to refer to the combination of any of postdictive completeness, conservativeness and prudence, just not all three.

Restrictions on TxtEx-Learning

- In Yoshinaka 2009 (a very nice paper) it is claimed that the following three restrictions force fairness of polynomial time restricted learning:
- postdictive completeness a.k.a. consistency (i.e., a learner only outputs hypotheses postdicting all known data);
- conservativeness (i.e., a learner revises its hypothesis only when that hypothesis fails to predict a new, current datum) and
- prudence (i.e., a learner's hypotheses are only for languages it can learn).
- Below, we will talk about 3-of-3 to refer to the combination of postdictive completeness, conservativeness and prudence.
- We will talk about <3 to refer to the combination of any of postdictive completeness, conservativeness and prudence, just not all three.

Restrictions on TxtEx-Learning

- In Yoshinaka 2009 (a very nice paper) it is claimed that the following three restrictions force fairness of polynomial time restricted learning:
- postdictive completeness a.k.a. consistency (i.e., a learner only outputs hypotheses postdicting all known data);
- conservativeness (i.e., a learner revises its hypothesis only when that hypothesis fails to predict a new, current datum) and
- prudence (i.e., a learner's hypotheses are only for languages it can learn).
- Below, we will talk about 3-of-3 to refer to the combination of postdictive completeness, conservativeness and prudence.
- We will talk about <3 to refer to the combination of any of postdictive completeness, conservativeness and prudence, just not all three.

Restrictions on TxtEx-Learning

- In Yoshinaka 2009 (a very nice paper) it is claimed that the following three restrictions force fairness of polynomial time restricted learning:
- postdictive completeness a.k.a. consistency (i.e., a learner only outputs hypotheses postdicting all known data);
- conservativeness (i.e., a learner revises its hypothesis only when that hypothesis fails to predict a new, current datum) and
- prudence (i.e., a learner's hypotheses are only for languages it can learn).
- Below, we will talk about 3-of-3 to refer to the combination of postdictive completeness, conservativeness and prudence.
- We will talk about <3 to refer to the combination of any of postdictive completeness, conservativeness and prudence, just not all three.

Restrictions on TxtEx-Learning

- In Yoshinaka 2009 (a very nice paper) it is claimed that the following three restrictions force fairness of polynomial time restricted learning:
- postdictive completeness a.k.a. consistency (i.e., a learner only outputs hypotheses postdicting all known data);
- conservativeness (i.e., a learner revises its hypothesis only when that hypothesis fails to predict a new, current datum) and
- prudence (i.e., a learner's hypotheses are only for languages it can learn).
- Below, we will talk about 3-of-3 to refer to the combination of postdictive completeness, conservativeness and prudence.
- We will talk about <3 to refer to the combination of any of postdictive completeness, conservativeness and prudence, just not all three.

Formally: Restrictions on TxtEx-Learning

- A TxtEx-learner h is called postdictively complete a.k.a. consistent iff, for all T, k, $h(T[k])$ correctly postdicts $T[k]$ $(= T(0), \ldots, T(k-1))$, i.e.,

$$\{T(0), \ldots, T(k-1)\} \subseteq V_{h(T[k])}$$

  ($V_{h(T[k])}$ is the language computed by the program $h(T[k])$).
- A TxtEx-learner h is called conservative iff, for all T, k, if $h(T[k+1]) \neq h(T[k])$, then

$$\{T(0), \ldots, T(k)\} \not\subseteq V_{h(T[k])}.$$

- A TxtEx-learner h is called prudent iff, for all T, k, $V_{h(T[k])}$ is TxtEx-learnable by h.

Formally: Restrictions on TxtEx-Learning

- A TxtEx-learner h is called postdictively complete a.k.a. consistent iff, for all T, k, $h(T[k])$ correctly postdicts $T[k]$ $(= T(0), \ldots, T(k-1))$, i.e.,

$$\{T(0), \ldots, T(k-1)\} \subseteq V_{h(T[k])}$$

  ($V_{h(T[k])}$ is the language computed by the program $h(T[k])$).

- A TxtEx-learner h is called conservative iff, for all T, k, if $h(T[k+1]) \neq h(T[k])$, then

$$\{T(0), \ldots, T(k)\} \not\subseteq V_{h(T[k])}.$$

- A TxtEx-learner h is called prudent iff, for all T, k, $V_{h(T[k])}$ is TxtEx-learnable by h.

Formally: Restrictions on TxtEx-Learning

- A TxtEx-learner h is called postdictively complete a.k.a. consistent iff, for all T, k, $h(T[k])$ correctly postdicts $T[k]$ $(= T(0), \dots, T(k-1))$, i.e.,

$$\{T(0), \dots, T(k-1)\} \subseteq V_{h(T[k])}$$

($V_{h(T[k])}$ is the language computed by the program $h(T[k])$).

- A TxtEx-learner h is called conservative iff, for all T, k, if $h(T[k+1]) \neq h(T[k])$, then

$$\{T(0), \dots, T(k)\} \not\subseteq V_{h(T[k])}.$$

- A TxtEx-learner h is called prudent iff, for all T, k, $V_{h(T[k])}$ is TxtEx-learnable by h.

Formally: Restrictions on TxtEx-Learning

- A TxtEx-learner h is called postdictively complete a.k.a. consistent iff, for all T, k, $h(T[k])$ correctly postdicts $T[k]$ $(= T(0), \ldots, T(k-1))$, i.e.,

$$\{T(0), \ldots, T(k-1)\} \subseteq V_{h(T[k])}$$

  ($V_{h(T[k])}$ is the language computed by the program $h(T[k])$).
- A TxtEx-learner h is called conservative iff, for all T, k, if $h(T[k+1]) \neq h(T[k])$, then

$$\{T(0), \ldots, T(k)\} \not\subseteq V_{h(T[k])}.$$

- A TxtEx-learner h is called prudent iff, for all T, k, $V_{h(T[k])}$ is TxtEx-learnable by h.

Uniformly Polynomial Time Decidable Hypothesis Spaces

- We call an hypothesis space V uniformly polynomial time decidable iff there is a polynomial time computable function which, given x, e, decides whether $x \in V_e$.
- Recall that Yoshinaka 2009 claims that 3-of-3 suffices to forbid all Pitt-style delaying tricks.
- For such uniformly polynomial time decidable hypothesis spaces (with a few easy closure properties) we get the strongest possible refutation of Yoshinaka's Thesis:
- Each set of languages 3-of-3 TxtEx-learnable wrt V is so learnable in polynomial time (by means of delaying tricks).
- Further, each set of languages <3 TxtEx-learnable wrt V is so learnable in polynomial time (by means of delaying tricks).
- Hence, with each combination of restrictions, we get arbitrary delaying tricks.

Uniformly Polynomial Time Decidable Hypothesis Spaces

- We call an hypothesis space V uniformly polynomial time decidable iff there is a polynomial time computable function which, given x, e, decides whether $x \in V_e$.
- Recall that Yoshinaka 2009 claims that 3-of-3 suffices to forbid all Pitt-style delaying tricks.
- For such uniformly polynomial time decidable hypothesis spaces (with a few easy closure properties) we get the strongest possible refutation of Yoshinaka's Thesis:
- Each set of languages 3-of-3 TxtEx-learnable wrt V is so learnable in polynomial time (by means of delaying tricks).
- Further, each set of languages <3 TxtEx-learnable wrt V is so learnable in polynomial time (by means of delaying tricks).
- Hence, with each combination of restrictions, we get arbitrary delaying tricks.

Uniformly Polynomial Time Decidable Hypothesis Spaces

- We call an hypothesis space V uniformly polynomial time decidable iff there is a polynomial time computable function which, given x, e, decides whether $x \in V_e$.
- Recall that Yoshinaka 2009 claims that 3-of-3 suffices to forbid all Pitt-style delaying tricks.
- For such uniformly polynomial time decidable hypothesis spaces (with a few easy closure properties) we get the strongest possible refutation of Yoshinaka's Thesis:
- Each set of languages 3-of-3 TxtEx-learnable wrt V is so learnable in polynomial time (by means of delaying tricks).
- Further, each set of languages <3 TxtEx-learnable wrt V is so learnable in polynomial time (by means of delaying tricks).
- Hence, with each combination of restrictions, we get arbitrary delaying tricks.

Uniformly Polynomial Time Decidable Hypothesis Spaces

- We call an hypothesis space V uniformly polynomial time decidable iff there is a polynomial time computable function which, given x, e, decides whether $x \in V_e$.
- Recall that Yoshinaka 2009 claims that 3-of-3 suffices to forbid all Pitt-style delaying tricks.
- For such uniformly polynomial time decidable hypothesis spaces (with a few easy closure properties) we get the strongest possible refutation of Yoshinaka's Thesis:
- Each set of languages 3-of-3 TxtEx-learnable wrt V is so learnable in polynomial time (by means of delaying tricks).
- Further, each set of languages <3 TxtEx-learnable wrt V is so learnable in polynomial time (by means of delaying tricks).
- Hence, with each combination of restrictions, we get arbitrary delaying tricks.

Uniformly Polynomial Time Decidable Hypothesis Spaces

- We call an hypothesis space V uniformly polynomial time decidable iff there is a polynomial time computable function which, given x, e, decides whether $x \in V_e$.
- Recall that Yoshinaka 2009 claims that 3-of-3 suffices to forbid all Pitt-style delaying tricks.
- For such uniformly polynomial time decidable hypothesis spaces (with a few easy closure properties) we get the strongest possible refutation of Yoshinaka's Thesis:
- Each set of languages 3-of-3 TxtEx-learnable wrt V is so learnable in polynomial time (by means of delaying tricks).
- Further, each set of languages <3 TxtEx-learnable wrt V is so learnable in polynomial time (by means of delaying tricks).
- Hence, with each combination of restrictions, we get arbitrary delaying tricks.

Uniformly Polynomial Time Decidable Hypothesis Spaces

- We call an hypothesis space V uniformly polynomial time decidable iff there is a polynomial time computable function which, given x, e, decides whether $x \in V_e$.
- Recall that Yoshinaka 2009 claims that 3-of-3 suffices to forbid all Pitt-style delaying tricks.
- For such uniformly polynomial time decidable hypothesis spaces (with a few easy closure properties) we get the strongest possible refutation of Yoshinaka's Thesis:
- Each set of languages 3-of-3 TxtEx-learnable wrt V is so learnable in polynomial time (by means of delaying tricks).
- Further, each set of languages <3 TxtEx-learnable wrt V is so learnable in polynomial time (by means of delaying tricks).
- Hence, with each combination of restrictions, we get arbitrary delaying tricks.

Uniformly Polynomial Time Decidable Hypothesis Spaces

- We call an hypothesis space V uniformly polynomial time decidable iff there is a polynomial time computable function which, given x, e, decides whether $x \in V_e$.
- Recall that Yoshinaka 2009 claims that 3-of-3 suffices to forbid all Pitt-style delaying tricks.
- For such uniformly polynomial time decidable hypothesis spaces (with a few easy closure properties) we get the strongest possible refutation of Yoshinaka's Thesis:
- Each set of languages 3-of-3 TxtEx-learnable wrt V is so learnable in polynomial time (by means of delaying tricks).
- Further, each set of languages <3 TxtEx-learnable wrt V is so learnable in polynomial time (by means of delaying tricks).
- Hence, with each combination of restrictions, we get arbitrary delaying tricks.

Uniformly Decidable Hypothesis Spaces

- We call an hypothesis space V uniformly decidable iff there is a computable function which, given x, e, decides whether $x \in V_e$.
- There are uniformly decidable V such that the set of graphs of all linear time computable functions is TxtEx-learnable wrt V by a polynomial time computable learner observing 3-of-3 – and, importantly, our proof uses Pitt-style delaying.
- Hence, not all delaying tricks are forbidden.
- Furthermore, there are uniformly decidable V such that the set of all graphs of exponential time computable functions is TxtEx-learnable wrt V by a computable learner observing postdictive completeness, but it is not so learnable by a polynomial time computable learner.
- Hence, some delaying is forbidden with postd. completeness.
- However, for all combinations of <3 which do not involve postdictive completeness, arbitrary delaying is possible.

Uniformly Decidable Hypothesis Spaces

- We call an hypothesis space V uniformly decidable iff there is a computable function which, given x, e, decides whether $x \in V_e$.
- There are uniformly decidable V such that the set of graphs of all linear time computable functions is TxtEx-learnable wrt V by a polynomial time computable learner observing 3-of-3 – and, importantly, our proof uses Pitt-style delaying.
- Hence, not all delaying tricks are forbidden.
- Furthermore, there are uniformly decidable V such that the set of all graphs of exponential time computable functions is TxtEx-learnable wrt V by a computable learner observing postdictive completeness, but it is not so learnable by a polynomial time computable learner.
- Hence, some delaying is forbidden with postd. completeness.
- However, for all combinations of <3 which do not involve postdictive completeness, arbitrary delaying is possible.

Uniformly Decidable Hypothesis Spaces

- We call an hypothesis space V uniformly decidable iff there is a computable function which, given x, e, decides whether $x \in V_e$.
- There are uniformly decidable V such that the set of graphs of all linear time computable functions is TxtEx-learnable wrt V by a polynomial time computable learner observing 3-of-3 – and, importantly, our proof uses Pitt-style delaying.
- Hence, not all delaying tricks are forbidden.
- Furthermore, there are uniformly decidable V such that the set of all graphs of exponential time computable functions is TxtEx-learnable wrt V by a computable learner observing postdictive completeness, but it is not so learnable by a polynomial time computable learner.
- Hence, some delaying is forbidden with postd. completeness.
- However, for all combinations of <3 which do not involve postdictive completeness, arbitrary delaying is possible.

Uniformly Decidable Hypothesis Spaces

- We call an hypothesis space V uniformly decidable iff there is a computable function which, given x, e, decides whether $x \in V_e$.
- There are uniformly decidable V such that the set of graphs of all linear time computable functions is TxtEx-learnable wrt V by a polynomial time computable learner observing 3-of-3 – and, importantly, our proof uses Pitt-style delaying.
- Hence, not all delaying tricks are forbidden.
- Furthermore, there are uniformly decidable V such that the set of all graphs of exponential time computable functions is TxtEx-learnable wrt V by a computable learner observing postdictive completeness, but it is not so learnable by a polynomial time computable learner.
- Hence, some delaying is forbidden with postd. completeness.
- However, for all combinations of <3 which do not involve postdictive completeness, arbitrary delaying is possible.

Uniformly Decidable Hypothesis Spaces

- We call an hypothesis space V uniformly decidable iff there is a computable function which, given x, e, decides whether $x \in V_e$.
- There are uniformly decidable V such that the set of graphs of all linear time computable functions is TxtEx-learnable wrt V by a polynomial time computable learner observing 3-of-3 – and, importantly, our proof uses Pitt-style delaying.
- Hence, not all delaying tricks are forbidden.
- Furthermore, there are uniformly decidable V such that the set of all graphs of exponential time computable functions is TxtEx-learnable wrt V by a computable learner observing postdictive completeness, but it is not so learnable by a polynomial time computable learner.
- Hence, some delaying is forbidden with postd. completeness.
- However, for all combinations of <3 which do not involve postdictive completeness, arbitrary delaying is possible.

Uniformly Decidable Hypothesis Spaces

- We call an hypothesis space V uniformly decidable iff there is a computable function which, given x, e, decides whether $x \in V_e$.
- There are uniformly decidable V such that the set of graphs of all linear time computable functions is TxtEx-learnable wrt V by a polynomial time computable learner observing 3-of-3 – and, importantly, our proof uses Pitt-style delaying.
- Hence, not all delaying tricks are forbidden.
- Furthermore, there are uniformly decidable V such that the set of all graphs of exponential time computable functions is TxtEx-learnable wrt V by a computable learner observing postdictive completeness, but it is not so learnable by a polynomial time computable learner.
- Hence, some delaying is forbidden with postd. completeness.
- However, for all combinations of <3 which do not involve postdictive completeness, arbitrary delaying is possible.

Uniformly Decidable Hypothesis Spaces

- We call an hypothesis space V uniformly decidable iff there is a computable function which, given x, e, decides whether $x \in V_e$.
- There are uniformly decidable V such that the set of graphs of all linear time computable functions is TxtEx-learnable wrt V by a polynomial time computable learner observing 3-of-3 – and, importantly, our proof uses Pitt-style delaying.
- Hence, not all delaying tricks are forbidden.
- Furthermore, there are uniformly decidable V such that the set of all graphs of exponential time computable functions is TxtEx-learnable wrt V by a computable learner observing postdictive completeness, but it is not so learnable by a polynomial time computable learner.
- Hence, some delaying is forbidden with postd. completeness.
- However, for all combinations of <3 which do not involve postdictive completeness, arbitrary delaying is possible.

Thank you.