Learning Binary Relations

Presented by Alan Duan

Motivation of Binary Relations

Let's start by considering the set of all students (let's call it S), and the set of all topics in this course (T).

Motivation of Binary Relations

Let's start by considering the set of all students (let's call it S), and the set of all topics in this course (T).

S and T are **related** by some rule.

Motivation of Binary Relations

Let's start by considering the set of all students (let's call it S), and the set of all topics in this course (T).

S and T are **related** by some rule.

Consider one relation: Student s presents topic t.

Motivation of Binary Relations

Let's start by considering the set of all students (let's call it S), and the set of all topics in this course (T).

S and T are **related** by some rule.

Consider one relation: Student s presents topic t.

For example, Alan presents the topic '*learning binary relations*', and Mark presented both '*tail inequalities*' and '*realizable selective sampling*'.

Motivation of Binary Relations

Let's start by considering the set of all students (let's call it S), and the set of all topics in this course (T).

S and T are **related** by some rule.

Consider one relation: Student s presents topic t.

For example, Alan presents the topic '*learning binary relations*', and Mark presented both '*tail inequalities*' and '*realizable selective sampling*'.

Clearly, student s either presents topic t, or does not.

Motivation of Binary Relations

Let's start by considering the set of all students (let's call it S), and the set of all topics in this course (T).

S and T are **related** by some rule.

Consider one relation: Student s presents topic t.

For example, Alan presents the topic '*learning binary relations*', and Mark presented both '*tail inequalities*' and '*realizable selective sampling*'.

Clearly, student s either presents topic t, or does not.

The predicate relating the two sets of variables is either true or false.

Motivation of Binary Relations

Let's start by considering the set of all students (let's call it S), and the set of all topics in this course (T).

S and T are **related** by some rule.

Consider one relation: Student s presents topic t.

For example, Alan presents the topic '*learning binary relations*', and Mark presented both '*tail inequalities*' and '*realizable selective sampling*'.

Clearly, student s either presents topic t, or does not.

The predicate relating the two sets of variables is either true or false.

We call this a **binary relation**.

Formal Definition of Binary Relations

A binary relation R between two sets A and B is a subset of $A \times B$.

Formal Definition of Binary Relations

A binary relation R between two sets A and B is a subset of $A \times B$.

Each binary relation is associated with a predicate $P : A \times B \mapsto \{0, 1\}$:

Formal Definition of Binary Relations

A binary relation R between two sets A and B is a subset of $A \times B$.

Each binary relation is associated with a predicate $P : A \times B \mapsto \{0, 1\}$:

$$P(a, b) = \begin{cases} 1, & \text{if } (a, b) \in R \\ 0, & \text{otherwise} \end{cases}$$

Formal Definition of Binary Relations

A binary relation R between two sets A and B is a subset of $A \times B$.

Each binary relation is associated with a predicate $P : A \times B \mapsto \{0, 1\}$:

$$P(a, b) = \begin{cases} 1, & \text{if } (a, b) \in R \\ 0, & \text{otherwise} \end{cases}$$

Note :

1. Binary relations can be defined between different set (e.g.: Netflix user and movie), or the set with itself (e.g.: the relation 'divides' between $\mathbb{N}_+$ and $\mathbb{N}_+$).

Formal Definition of Binary Relations

A binary relation R between two sets A and B is a subset of $A \times B$.

Each binary relation is associated with a predicate $P : A \times B \mapsto \{0, 1\}$:

$$P(a, b) = \begin{cases} 1, & \text{if } (a, b) \in R \\ 0, & \text{otherwise} \end{cases}$$

Note :

1. Binary relations can be defined between different set (e.g.: Netflix user and movie), or the set with itself (e.g.: the relation 'divides' between $\mathbb{N}_+$ and $\mathbb{N}_+$).

2. In binary relations, the order matters.

Representing a Binary Relations

- $n \times m$ binary matrix

	Topics in Learning Theory	Machine Learning	Operating System
Alan	1	0	0
Bob	1	1	0
Cathy	0	0	1
David	0	0	0

Representing a Binary Relations

- $n \times m$ binary matrix

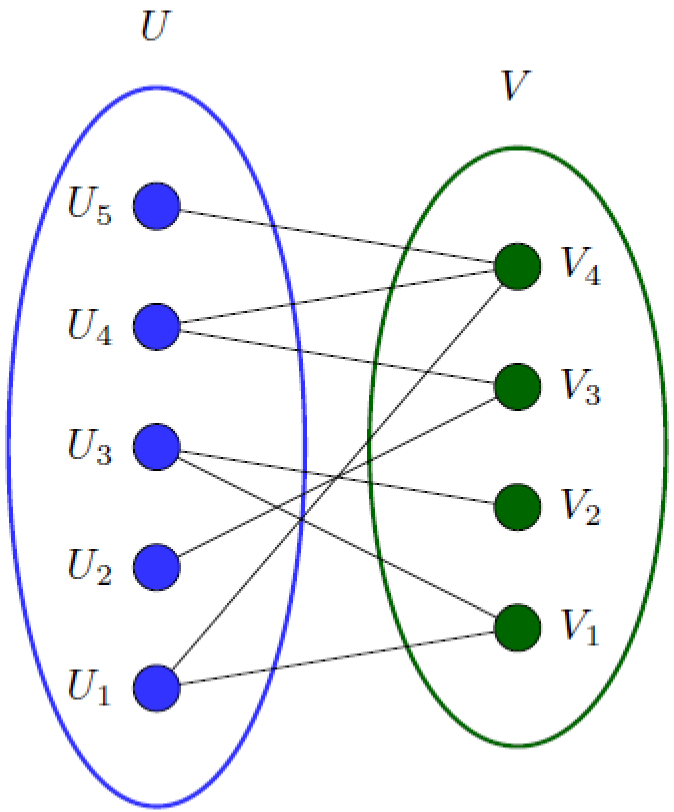
	Topics in Learning Theory	Machine Learning	Operating System
Alan	1	0	0
Bob	1	1	0
Cathy	0	0	1
David	0	0	0

- 2-column table

Student	**Course**
Alan	Topics in Learning Theory
Bob	Topics in Learning Theory
Bob	Machine Learning
Cathy	Operating System

Representing a Binary Relations (cont'd)

- Bipartite graph



Learning Binary Relations

Setting

We are learning binary relations between two set A and B represented by predicate P. Denote $|A| = n$ and $|B| = m$.

In each trial t:

- learner is given an unlabeled pair of object $x_t = (a_t, b_t)$, where $a_t \in A, b_t \in B$
- learner predicts $\hat{y}_t = 0$ or 1
- reveals the answer y_t
- if answer and prediction are different, record it as a *mistake*

Goal: Minimize the number of incorrect predictions

Learning Binary Relations

Question: Can we reduce the learning of binary relations to something we have seen?

Learning Binary Relations

Question: Can we reduce the learning of binary relations to something we have seen?

Yes!

Learning Binary Relations

Question: Can we reduce the learning of binary relations to something we have seen?

Yes!

- $\mathcal{X} = A \times B$, $\mathcal{Y} = \{0, 1\}$
- Target hypothesis $h = P$

This is an online concept learning (realizable) setting!

Learning Binary Relations

Question: Can we reduce the learning of binary relations to something we have seen?

Yes!

- $\mathcal{X} = A \times B$, $\mathcal{Y} = \{0, 1\}$
- Target hypothesis $h = P$

This is an online concept learning (realizable) setting!

Note :

1. In this presentation, we will use these notation from concept learning interchangably from time to time.
2. We will see what is special about learning binary relations in a bit!

Learning Binary Relations

A few more terms

Let $\mathcal{X}$ be a finite learning domain. Let C be a concept class over $\mathcal{X}$.

A learner is *consistent* if, on every trial, there exists some concept $c \in C$ such that:

$$c(x_k) = \begin{cases} \hat{y}_t, & \text{if } k = t \\ y_k, & \text{if } k = 1, \dots, t-1 \end{cases}$$

A *query sequence* $\pi = \langle x_1, x_2, \dots, x_{|\mathcal{X}|} \rangle$ is a permutation of $\mathcal{X}$, where $x_t \in \mathcal{X}$ is the instance presented to the learner at the t^{th} trial.

Learning Binary Relations

Who determines the query sequence?

Learning Binary Relations

Who determines the query sequence?

Director!

Learning Binary Relations

Who determines the query sequence?

Director!

In this presentation, we will consider the following settings:

- Director Agnostic: we want some mistake bounds regardless of the director.

Learning Binary Relations

Who determines the query sequence?

Director!

In this presentation, we will consider the following settings:

- Director Agnostic: we want some mistake bounds regardless of the director.
- Self-directed: the learner itself chooses π.

Learning Binary Relations

Who determines the query sequence?

Director!

In this presentation, we will consider the following settings:

- Director Agnostic: we want some mistake bounds regardless of the director.
- Self-directed: the learner itself chooses π.
- Teacher-directed: A teacher who knows the target relation and wants to minimize the learner's mistakes by choosing π; Teacher can choose x_t with the knowledge of 1) target relation, 2) $x_1, \ldots, x_{t-1}$, 3) $\hat{y}_1, \ldots, \hat{y}_{t-1}$.

Learning Binary Relations

Who determines the query sequence?

Director!

In this presentation, we will consider the following settings:

- Director Agnostic: we want some mistake bounds regardless of the director.
- Self-directed: the learner itself chooses π.
- Teacher-directed: A teacher who knows the target relation and wants to minimize the learner's mistakes by choosing π; Teacher can choose x_t with the knowledge of 1) target relation, 2) $x_1, \ldots, x_{t-1}$, 3) $\hat{y}_1, \ldots, \hat{y}_{t-1}$.
- Adversary-directed: An adversary who tries to maximize the learner's mistakes, knows the learner's algorithm and has unlimited computing power, chooses π.

Learning Binary Relations

Who determines the query sequence?

Director!

In this presentation, we will consider the following settings:

- Director Agnostic: we want some mistake bounds regardless of the director.
- Self-directed: the learner itself chooses π.
- Teacher-directed: A teacher who knows the target relation and wants to minimize the learner's mistakes by choosing π; Teacher can choose x_t with the knowledge of 1) target relation, 2) $x_1, \ldots, x_{t-1}$, 3) $\hat{y}_1, \ldots, \hat{y}_{t-1}$.
- Adversary-directed: An adversary who tries to maximize the learner's mistakes, knows the learner's algorithm and has unlimited computing power, chooses π.

For teacher-directed setting, we want to consider **worst case mistake bound** over all consistent learners. (why?)

Motivation of k-binary-relations

Now let's talk about what can be special about binary relations.

Motivation of k-binary-relations

Now let's talk about what can be special about binary relations.

1. There are two sets of objects (instead of one)
2. We are learning a relationship between the two sets (instead of some concepts for classification)

Motivation of k-binary-relations

Now let's talk about what can be special about binary relations.

1. There are two sets of objects (instead of one)
2. We are learning a relationship between the two sets (instead of some concepts for classification)

Then it's natural to impose some **structures** in the relation.

Motivation of k-binary-relations

Now let's talk about what can be special about binary relations.

1. There are two sets of objects (instead of one)
2. We are learning a relationship between the two sets (instead of some concepts for classification)

Then it's natural to impose some **structures** in the relation.

If there's no structure, we can't do any better than random guessing.

Motivation of k-binary-relations

Now let's talk about what can be special about binary relations.

1. There are two sets of objects (instead of one)
2. We are learning a relationship between the two sets (instead of some concepts for classification)

Then it's natural to impose some **structures** in the relation.

If there's no structure, we can't do any better than random guessing.

What can be a natural structure?

Motivation of k-binary-relations

Consider our example of "student presenting topic in this class" again.

Motivation of k-binary-relations

Consider our example of "student presenting topic in this class" again.

We know each student only presents at most once.

Motivation of k-binary-relations

Consider our example of "student presenting topic in this class" again.

We know each student only presents at most once.

Then if we want to learn which topic Alan presents, how many possibilities there are?

Motivation of k-binary-relations

Consider our example of "student presenting topic in this class" again.

We know each student only presents at most once.

Then if we want to learn which topic Alan presents, how many possibilities there are?

Equiv : If we represent this binary relation using a $n \times m$ matrix, how many possible *row type* could the row for Alan be?

	Splitting Index	Equivalence Queries	...	Leaderboard
Alan	?	?	$\ldots$	?

Motivation of k-binary-relations

Consider our example of "student presenting topic in this class" again.

We know each student only presents at most once.

Then if we want to learn which topic Alan presents, how many possibilities there are?

Equiv : If we represent this binary relation using a $n \times m$ matrix, how many possible *row type* could the row for Alan be?

	Splitting Index	Equivalence Queries	...	Leaderboard
$Alan$	?	?	$\dots$	?

First of all, it's a fixed number!

Motivation of k-binary-relations

Consider our example of "student presenting topic in this class" again.

We know each student only presents at most once.

Then if we want to learn which topic Alan presents, how many possibilities there are?

Equiv : If we represent this binary relation using a $n \times m$ matrix, how many possible *row type* could the row for Alan be?

	Splitting Index	Equivalence Queries	...	Leaderboard
$Alan$	$?$	$?$	$\dots$	$?$

First of all, it's a fixed number!

Second, it's way less than 2^m (where m is total number of topics)

Motivation of k-binary-relations

Consider our example of "student presenting topic in this class" again.

We know each student only presents at most once.

Then if we want to learn which topic Alan presents, how many possibilities there are?

Equiv : If we represent this binary relation using a $n \times m$ matrix, how many possible *row type* could the row for Alan be?

	Splitting Index	Equivalence Queries	...	Leaderboard
$Alan$	?	?	$\dots$	?

First of all, it's a fixed number!

Second, it's way less than 2^m (where m is total number of topics)

A little math tells us the answer is $m + 1$.

Motivation of k-binary-relations

Consider our example of "student presenting topic in this class" again.

We know each student only presents at most once.

Then if we want to learn which topic Alan presents, how many possibilities there are?

Equiv : If we represent this binary relation using a $n \times m$ matrix, how many possible *row type* could the row for Alan be?

	Splitting Index	Equivalence Queries	...	Leaderboard
$Alan$	?	?	$\dots$	?

First of all, it's a fixed number!

Second, it's way less than 2^m (where m is total number of topics)

A little math tells us the answer is $m + 1$.

We use k to represent the distinct row types in the matrix. We call this type of relation *k-binary-relations*.

General bounds applied to all directors

Theorem 1 (Lower Bound) For any $0 < \beta \leq 1$, any prediction algorithm makes at least $(1-\beta)km + n\lfloor \log(\beta k) \rfloor - (1-\beta)k\lfloor \log(\beta k) \rfloor$ mistakes regardless of the query sequence.

General bounds applied to all directors

Theorem 1 (Lower Bound) For any $0 < \beta \leq 1$, any prediction algorithm makes at least $(1-\beta)km + n\lfloor\log(\beta k)\rfloor - (1-\beta)k\lfloor\log(\beta k)\rfloor$ mistakes regardless of the query sequence.

Proof :

We prove the bound by showing that for any algorithm, there exists a matrix (filled by adversary) that forces the learner to make such number of mistakes.

General bounds applied to all directors

Theorem 1 (Lower Bound) For any $0 < \beta \leq 1$, any prediction algorithm makes at least $(1-\beta)km + n\lfloor \log(\beta k) \rfloor - (1-\beta)k\lfloor \log(\beta k) \rfloor$ mistakes regardless of the query sequence.

Proof :

We prove the bound by showing that for any algorithm, there exists a matrix (filled by adversary) that forces the learner to make such number of mistakes.

For entries in the first p columns, the adversary replies that the learner's prediction is incorrect.

For entries in the first q rows, the adversary also replies that the learner's prediction is incorrect.

General bounds applied to all directors

Theorem 1 (Lower Bound) For any $0 < \beta \leq 1$, any prediction algorithm makes at least $(1-\beta)km + n\lfloor \log(\beta k) \rfloor - (1-\beta)k\lfloor \log(\beta k) \rfloor$ mistakes regardless of the query sequence.

Proof :

We prove the bound by showing that for any algorithm, there exists a matrix (filled by adversary) that forces the learner to make such number of mistakes.

For entries in the first p columns, the adversary replies that the learner's prediction is incorrect.

For entries in the first q rows, the adversary also replies that the learner's prediction is incorrect.

Constraint for adversary: it cannot create too many row types.

General bounds applied to all directors

Theorem 1 (Lower Bound) For any $0 < \beta \leq 1$, any prediction algorithm makes at least $(1-\beta)km + n\lfloor \log(\beta k) \rfloor - (1-\beta)k\lfloor \log(\beta k) \rfloor$ mistakes regardless of the query sequence.

Proof :

We prove the bound by showing that for any algorithm, there exists a matrix (filled by adversary) that forces the learner to make such number of mistakes.

For entries in the first p columns, the adversary replies that the learner's prediction is incorrect.

For entries in the first q rows, the adversary also replies that the learner's prediction is incorrect.

Constraint for adversary: it cannot create too many row types.

By forcing mistakes in the first p columns, at most 2^p row types can be created.

By forcing mistakes in the first q rows, at most q row types can be created.

$$2^p + q = k$$

Proof (cont'd):

By forcing mistakes in the first p columns, at most 2^p row types can be created.

By forcing mistakes in the first q rows, at most q row types can be created.

$$2^p + q = k$$

Set $2^p = \beta k, q = (1-\beta)k$, we can get $p = \lfloor \log(\beta k) \rfloor$, $q = (1-\beta)k$.

The mistake bound: $(1-\beta)k \cdot m + \lfloor \log(\beta k) \rfloor \cdot n - (1-\beta)k \lfloor \log(\beta k) \rfloor$.

General bounds applied to all directors

Theorem 2 (Upper Bound) The halving algorithm achieves a $km + (n - k)\log k$ mistake bound.

General bounds applied to all directors

Theorem 2 (Upper Bound) The halving algorithm achieves a $km + (n-k)\log k$ mistake bound.

Proof :

We know halving algorithm makes at most $\log|C|$ mistakes. The question is what is $|C|$.

General bounds applied to all directors

Theorem 2 (Upper Bound) The halving algorithm achieves a $km + (n-k)\log k$ mistake bound.

Proof :

We know halving algorithm makes at most $\log |C|$ mistakes. The question is what is $|C|$.

We count how large C can be:

General bounds applied to all directors

Theorem 2 (Upper Bound) The halving algorithm achieves a $km + (n-k)\log k$ mistake bound.

Proof :

We know halving algorithm makes at most $\log |C|$ mistakes. The question is what is $|C|$.

We count how large C can be:

There are $(2^m)^k = 2^{km}$ ways to select k row types.

General bounds applied to all directors

Theorem 2 (Upper Bound) The halving algorithm achieves a $km + (n-k)\log k$ mistake bound.

Proof :

We know halving algorithm makes at most $\log |C|$ mistakes. The question is what is $|C|$.

We count how large C can be:

There are $(2^m)^k = 2^{km}$ ways to select k row types.

There are $k^{(n-k)}$ ways to assign one of the row types to each of the remaining $n-k$ rows.

General bounds applied to all directors

Theorem 2 (Upper Bound) The halving algorithm achieves a $km + (n-k)\log k$ mistake bound.

Proof :

We know halving algorithm makes at most $\log |C|$ mistakes. The question is what is $|C|$.

We count how large C can be:

There are $(2^m)^k = 2^{km}$ ways to select k row types.

There are $k^{(n-k)}$ ways to assign one of the row types to each of the remaining $n-k$ rows.

$|C| \leq 2^{km} k^{(n-k)}$.

General bounds applied to all directors

Theorem 2 (Upper Bound) The halving algorithm achieves a $km + (n-k)\log k$ mistake bound.

Proof :

We know halving algorithm makes at most $\log |C|$ mistakes. The question is what is $|C|$.

We count how large C can be:

There are $(2^m)^k = 2^{km}$ ways to select k row types.

There are $k^{(n-k)}$ ways to assign one of the row types to each of the remaining $n-k$ rows.

$|C| \leq 2^{km} k^{(n-k)}$.

$\log |C| \leq km + (n-k)\log k$.

General bounds applied to all directors

Theorem 2 (Upper Bound) The halving algorithm achieves a $km + (n-k)\log k$ mistake bound.

Proof :

We know halving algorithm makes at most $\log|C|$ mistakes. The question is what is $|C|$.

We count how large C can be:

There are $(2^m)^k = 2^{km}$ ways to select k row types.

There are $k^{(n-k)}$ ways to assign one of the row types to each of the remaining $n-k$ rows.

$|C| \leq 2^{km}k^{(n-k)}$.

$\log|C| \leq km + (n-k)\log k$.

Note : Halving algorithm (in general) can be computationally expensive!

Self-directed learning

Theorem 3 (Upper Bound) There exists an algorithm that achieves $km + (n - k)\lfloor \log k \rfloor$ mistake bound in self-directed learning setting.

Self-directed learning

Theorem 3 (Upper Bound) There exists an algorithm that achieves $km + (n-k)\lfloor \log k \rfloor$ mistake bound in self-directed learning setting.

Proof : We prove existence by showing one.

Self-directed learning

Theorem 3 (Upper Bound) There exists an algorithm that achieves $km + (n-k)\lfloor \log k \rfloor$ mistake bound in self-directed learning setting.

Proof : We prove existence by showing one.

Learner chooses to query row-by-row. Denote the learner's current estimate as $\hat{k}$. Initialize $\hat{k} = 1$.

Self-directed learning

Theorem 3 (Upper Bound) There exists an algorithm that achieves $km + (n-k)\lfloor \log k \rfloor$ mistake bound in self-directed learning setting.

Proof : We prove existence by showing one.

Learner chooses to query row-by-row. Denote the learner's current estimate as $\hat{k}$. Initialize $\hat{k} = 1$.

For the first row:

- Guess all entries. Record it as the first row type.

Self-directed learning

Theorem 3 (Upper Bound) There exists an algorithm that achieves $km + (n-k)\lfloor \log k \rfloor$ mistake bound in self-directed learning setting.

Proof : We prove existence by showing one.

Learner chooses to query row-by-row. Denote the learner's current estimate as $\hat{k}$. Initialize $\hat{k} = 1$.

For the first row:

- Guess all entries. Record it as the first row type.

For the rest rows:

- Predict row i, column j's value according to a majority vote of the recorded row templates that are consistent with row i
- If no such consistent template exists, guess all the rest entries in row i, and record it as a new type. $\hat{k} = \hat{k} + 1$.

How many mistakes have we made?

- For each new row template, we make at most m on each. Total is km.
- For each of the rest rows, we make at most $\lfloor \log \hat{k} \rfloor \leq \lfloor \log k \rfloor$ mistakes. The total is $(n-k)\lfloor \log k \rfloor$.
- Add up, we have the desired bound $km + (n-k)\lfloor \log k \rfloor$.

How many mistakes have we made?

- For each new row template, we make at most m on each. Total is km.
- For each of the rest rows, we make at most $\lfloor \log \hat{k} \rfloor \leq \lfloor \log k \rfloor$ mistakes. The total is $(n-k)\lfloor \log k \rfloor$.
- Add up, we have the desired bound $km + (n-k)\lfloor \log k \rfloor$.

Note :

1. Similar flavour as the halving algorithm -- but computationally tractable.

How many mistakes have we made?

- For each new row template, we make at most m on each. Total is km.
- For each of the rest rows, we make at most $\lfloor \log \hat{k} \rfloor \leq \lfloor \log k \rfloor$ mistakes. The total is $(n-k)\lfloor \log k \rfloor$.
- Add up, we have the desired bound $km + (n-k)\lfloor \log k \rfloor$.

Note :

1. Similar flavour as the halving algorithm -- but computationally tractable.
2. Do not need to know k a priori.

How many mistakes have we made?

- For each new row template, we make at most m on each. Total is km.
- For each of the rest rows, we make at most $\lfloor \log \hat{k} \rfloor \leq \lfloor \log k \rfloor$ mistakes. The total is $(n-k)\lfloor \log k \rfloor$.
- Add up, we have the desired bound $km + (n-k)\lfloor \log k \rfloor$.

Note :

1. Similar flavour as the halving algorithm -- but computationally tractable.
2. Do not need to know k a priori.
3. This bound is within a constant factor of the general lower bound (Theorem 1).

Teacher-directed

Theorem 4 (Upper Bound) The number of mistakes made with a helpful teacher as the director is at most $km + (n - k)(k - 1)$.

Teacher-directed

Theorem 4 (Upper Bound) The number of mistakes made with a helpful teacher as the director is at most $km + (n - k)(k - 1)$.

Proof:

First, the teacher presents the learner with one row of each type.

Teacher-directed

Theorem 4 (Upper Bound) The number of mistakes made with a helpful teacher as the director is at most $km + (n - k)(k - 1)$.

Proof:

First, the teacher presents the learner with one row of each type.

Then, for the rest of $(n - k)$ rows, the teacher presents $(k - 1)$ entries to distinguish it from the incorrect row types.

After this, for the rest of $(n - k)$ rows, its row type can be uniquely identified, and no more mistakes will be made.

Teacher-directed

Theorem 4 (Upper Bound) The number of mistakes made with a helpful teacher as the director is at most $km + (n-k)(k-1)$.

Proof:

First, the teacher presents the learner with one row of each type.

Then, for the rest of $(n-k)$ rows, the teacher presents $(k-1)$ entries to distinguish it from the incorrect row types.

After this, for the rest of $(n-k)$ rows, its row type can be uniquely identified, and no more mistakes will be made.

In total, the learner makes at most $km + (n-k)(k-1)$ mistakes.

Teacher-directed

Theorem 5 (Lower Bound) The number of mistakes made with a helpful teacher as the director is at least $\min\{nm, km + (n-k)(k-1)\}$.

Proof:

$$\begin{array}{r}
\text{5 row types}\left\{\begin{matrix}
0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0
\end{matrix}\right.\\
\begin{matrix}
0&0&0&0&0&0&0&0&0\\
&&&&\vdots&&&&\\
0&0&0&0&0&0&0&0&0
\end{matrix}
\end{array}$$

For the first k rows, they are of different row type. km mistakes are made.

Proof:

$$\begin{array}{cc}
\text{5 row types}\left\{\vphantom{\begin{matrix}0\\0\\0\\0\\0\end{matrix}}\right. & \begin{matrix}
0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0
\end{matrix}\\
 & \begin{matrix}
0&0&0&0&0&0&0&0&0\\
 & & & &\vdots& & & & \\
0&0&0&0&0&0&0&0&0
\end{matrix}
\end{array}$$

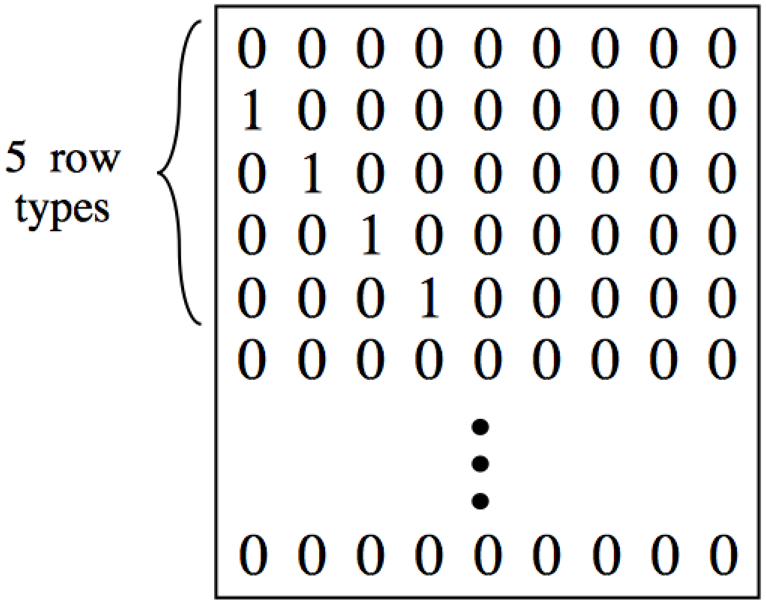
For the first k rows, they are of different row type. km mistakes are made.

For the rest of the rows:

When $(m+1) \geq k$: we need to know all first $k-1$ columns to uniquely identify the row type.

When $(m+1) < k$: we need to know all m columns to uniquely identify the row type.

Proof:



For the first k rows, they are of different row type. km mistakes are made.

For the rest of the rows:

When $(m+1) \geq k$: we need to know all first $k-1$ columns to uniquely identify the row type.

When $(m+1) < k$: we need to know all m columns to uniquely identify the row type.

Adding up, the mistake bound is $\min\{km + (n-k)m, km + (n-k)(k-1)\}$.

Teacher-directed

Question: Recall that the mistake bound for learner director is $km + (n-k)\lfloor \log k \rfloor$ **, while teacher-directed bound is** $km + (n-k)(k-1)$ **. Why is it even worse?**

Teacher-directed

Question: Recall that the mistake bound for learner director is $km + (n-k)\lfloor \log k \rfloor$, while teacher-directed bound is $km + (n-k)(k-1)$. Why is it even worse?

Teacher-directed case apply to all consistent learners!

Teacher-directed

Question: Recall that the mistake bound for learner director is $km + (n-k)\lfloor \log k \rfloor$, while teacher-directed bound is $km + (n-k)(k-1)$. Why is it even worse?

Teacher-directed case apply to all consistent learners!

A consistent learner may do *minority-vote* instead of *majority-vote*.

Adversary-directed

Theorem 6 (Lower Bound) Any prediction algorithm makes at least $\min\{nm, km + (n-k)\lfloor \log k \rfloor\}$ mistakes against an adversary-selected query sequence.

Adversary-directed

Theorem 6 (Lower Bound) Any prediction algorithm makes at least $\min\{nm, km + (n-k)\lfloor \log k \rfloor\}$ mistakes against an adversary-selected query sequence.

Proof:

The high level idea is to do the reverse of what the helpful teacher does -- try not to reveal the full information of row types!

Adversary-directed

Theorem 6 (Lower Bound) Any prediction algorithm makes at least $\min\{nm, km + (n-k)\lfloor \log k \rfloor\}$ mistakes against an adversary-selected query sequence.

Proof:

The high level idea is to do the reverse of what the helpful teacher does -- try not to reveal the full information of row types!

First, the adversary presents entries in the first $\min\{m, \lfloor \log k \rfloor\}$ columns for all n rows, and replies with each prediction is incorrect.

Adversary-directed

Theorem 6 (Lower Bound) Any prediction algorithm makes at least $\min\{nm, km + (n-k)\lfloor \log k \rfloor\}$ mistakes against an adversary-selected query sequence.

Proof:

The high level idea is to do the reverse of what the helpful teacher does -- try not to reveal the full information of row types!

First, the adversary presents entries in the first $\min\{m, \lfloor \log k \rfloor\}$ columns for all n rows, and replies with each prediction is incorrect.

Second, if $m > \lfloor \log k \rfloor$, the adversary presents remaining $m - \lfloor \log k \rfloor$ columns for each of the k row type, and forces mistakes on all of them.

Adversary-directed

Theorem 6 (Lower Bound) Any prediction algorithm makes at least $\min\{nm, km + (n-k)\lfloor \log k \rfloor\}$ mistakes against an adversary-selected query sequence.

Proof:

The high level idea is to do the reverse of what the helpful teacher does -- try not to reveal the full information of row types!

First, the adversary presents entries in the first $\min\{m, \lfloor \log k \rfloor\}$ columns for all n rows, and replies with each prediction is incorrect.

Second, if $m > \lfloor \log k \rfloor$, the adversary presents remaining $m - \lfloor \log k \rfloor$ columns for each of the k row type, and forces mistakes on all of them.

Adding up the number of mistakes, we get the desired bound.

Adversary-directed

How about upper bound?

Adversary-directed

How about upper bound?

Recall that if efficiency is not a concern, we can always run halving algorithm to get an upper bound of $km + (n-k)\lfloor \log k \rfloor$.

Adversary-directed

How about upper bound?

Recall that if efficiency is not a concern, we can always run halving algorithm to get an upper bound of $km + (n-k)\lfloor \log k \rfloor$.

If efficiency *is* a concern…let's start by considering a smaller k.

Adversary-directed

How about upper bound?

Recall that if efficiency is not a concern, we can always run halving algorithm to get an upper bound of $km + (n-k)\lfloor \log k \rfloor$.

If efficiency *is* a concern…let's start by considering a smaller k.

For $k = 1$, we are fine. Can achieve at most m mistakes.

Adversary-directed

How about upper bound?

Recall that if efficiency is not a concern, we can always run halving algorithm to get an upper bound of $km + (n-k)\lfloor \log k \rfloor$.

If efficiency *is* a concern…let's start by considering a smaller k.

For $k = 1$, we are fine. Can achieve at most m mistakes.

How about $k = 2$?

Adversary-directed

Theorem 7 (Upper Bound when k=2) There exists a polynomial prediction algorithm that makes at most $2m + n - 2$ mistakes against adversary-selected query sequence when $k = 2$.

Adversary-directed

Theorem 7 (Upper Bound when k=2) There exists a polynomial prediction algorithm that makes at most $2m + n - 2$ mistakes against adversary-selected query sequence when $k = 2$.

Proof: Let's do it on board!

Adversary-directed

How about $k \geq 3$?

Adversary-directed

How about $k \geq 3$?

We don't know!

Adversary-directed

How about $k \geq 3$?

We don't know!

To find if there's a matrix with at most k row types that is consistent with a partially known matrix M, is *NP-complete*.

Adversary-directed

How about $k \geq 3$?

We don't know!

To find if there's a matrix with at most k row types that is consistent with a partially known matrix M, is *NP-complete*.

To have a polynomial-time k-colorability oracle, we need to prove *P=NP*.

Adversary-directed

How about $k \geq 3$?

We don't know!

To find if there's a matrix with at most k row types that is consistent with a partially known matrix M, is *NP-complete*.

To have a polynomial-time k-colorability oracle, we need to prove *P=NP*.

This is left as an exercise.

Conclusion and Takeaways

1. In previous lectures, we usually focus on learner's algorithm, and assume the environment (director) as the worst case (adversary). It turns out to be not true in many real life cases. Maybe the director is trying to help learner to learn. And in those cases, we can indeed improve learner's performance.

Conclusion and Takeaways

1. In previous lectures, we usually focus on learner's algorithm, and assume the environment (director) as the worst case (adversary). It turns out to be not true in many real life cases. Maybe the director is trying to help learner to learn. And in those cases, we can indeed improve learner's performance.

2. It may be interesting to consider other "structures" in the learning setting. It may also be interesting to see how the results extend to k-ary relations.

Conclusion and Takeaways

1. In previous lectures, we usually focus on learner's algorithm, and assume the environment (director) as the worst case (adversary). It turns out to be not true in many real life cases. Maybe the director is trying to help learner to learn. And in those cases, we can indeed improve learner's performance.

2. It may be interesting to consider other "structures" in the learning setting. It may also be interesting to see how the results extend to k-ary relations.

3. Some learning on proof technique: to prove an upper bound, we can prove by showing an algorithm that satisfies the bound; to prove a lower bound, we can prove by showing there exists an adversary setting that all algorithms make at least this amount of mistake.

Reference

1. Learning Binary Relations and Total Orders, *Sally A. Goldman, Ronald L. Rivest, and Robert E. Schapire*, SIAM Journal on Computing 1993 22:5, 1006-1034.