COMS 6998-4 Fall 2017 Presenter: Yanlin Duan
October 25, 2017 Scribe: Yiyang Li

Learning Binary Relations

1 Introduction

In former lectures, we have learned a lot about online learning. The basic idea is to keep a subset of hypothesis space as the version space and reduce the version space by new data or queries. And we consider the data in any arbitrary form, which means we don't have any specific hypothesis on data's schema itself. Although it can be generalized easily, still we want to make a practical effort for a real problem. Now we are considering learning a special kind of concept–binary relations.

1.1 Motivation example

Considering the set of all students S and the set of all topics in course T. S and T are related by some rules. In this specific term, we may consider which course. has been presented by a certain student. It was clear that *present* it's a binary relation that defined on the S and T. We can also think of a predicate that takes one member from each set, and give the value of true or false, where true denotes that the two object does have the relation.

1.2 Formal Definition

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 $P : A \times B \mapsto \{0, 1\}$.

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

Also, we must note that while a binary relation can be defined on two different sets, such as Netflix users and the movies, it can be defined on a set with itself as well. The simple example can be the *divide* relation within $\mathbb{N}_+$.

2 Notation and Preliminaries

2.1 Representations

Now we are introducing the representation for a binary relation. There may exists several different ways that can do the job.

2.1.1 Matrix Represent

For two set A and B with respect to $|A| = n, |B| = m$, we define an $n \times m$ binary matrix M. Each entry of M can be either 0 or 1. Where 1 means two corresponding items has the relation and 0 means the opposite. Like in table 1, the matrix represents the course that the student takes in this semester. And we can see Alan takes only Learning Theory while David doesn't take any courses.

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

Table 1: *Take* relation between students and courses.

2.1.2 Two-column Table

This kind of representation is kind of trivial. We just need to save all entries which have value of 1 into a 2-column table. Table 2 shows the same relation for students and courses in table 1 as an example.

student	course
Alan	Learning Theory
Bob	Learning Theory
Bob	Machine Learning
Cathy	Operating System

Table 2: *Take* relation exact the same as table 1

2.1.3 Bipartite Graph

As for a graph, it is intuitive to use a bipartite graph as a representative for binary relations. And the transform from both sides is simple. You can see from figure 1 for an illustration of a bipartite graph in representing binary relations. The vertices represent the items in each set, while the edges denote the existing relations.

2.2 Learning Problem for Binary Relations

2.2.1 Definition

Now we are trying to give the definition for Binary relation learning. Assuming we learned 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 data pair $x_t = (a_t, b_t)$, where $a_t \in A$ and $b_t \in B$;
- learner predicts $\hat{y}_t = 0$ or 1;
- teacher reveal the answer y_t;
- if $\hat{y}_t \neq y_t$, record as a *mistake*.

The goal of binary relation learning is to reduce the number of mistakes the learner made before finally learned the complete relation. This setting is kind of like the Online Learning we learned before. If we see $x_t = (a_t, b_t)$ as one thing and target hypothesis is the predicate P, then we can convert a binary relation learning problem into a typical online learning problem in realizable setting and apply some algorithms from online learning. However, we are not doing that because a binary relation has its unique feature that can be used to improve the performance.

2.2.2 Few More

We say a learner is *consistent* if on every trial t, there exist some concept $c \in \mathcal{C}$,that

$$c(x_k) = \begin{cases} \hat{y_k}, & \text{if } k = t \\ y_k, & \text{o.w.} \end{cases}$$

And we consider a *query sequence* $\pi = \langle x_1, x_2, \ldots, x_{|\mathcal{X}|} \rangle$ as a permutation of all element of instance set $\mathcal{X}$.

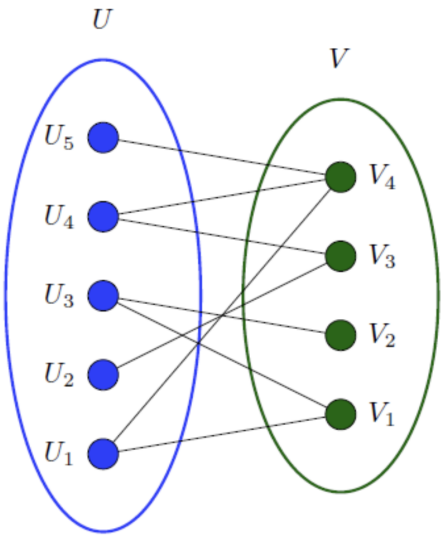

Figure 1: Bipartite Graph Example

2.2.3 Motivation of k-binary-relations

We must keep in mind why we are giving so many extra definitions along with the typical online learning. The binary relation can be special lying on several reasons.

1. We have two sets of objects while the normal online learning draws samples from one.
2. We learn the relations between this two sets instead of the concept of classification.

And while we are considering the relation between two sets, we don't want to have a random choice of which item will have relations with an arbitrary item in the other set. We may want to have this assumption that there only exists a type of relations between two sets which will be much less than 2^m(m will be the total number of columns in matrix representation). We use k to denote the total number of types of the column. We call this type of binary relation *k-binary-relations*.

3 Different Directors

In matrix representing, the learner asks for an arbitrary entry's value from teacher each time. The learner will give a prediction then receive the answer. This procedure can be seen as a sequence of positions, which we call *query sequence*.

After giving the whole definition, we can separate the problem by the different setting of *director*. A director decides the query sequence of the learning procedure in its own purpose. We have a different kind of director, and we may consider the problem in several settings.

1. **Director Agnostic**: We don't care about a specific director and want to get general bounds.
2. **Self-directed**: The learner chooses the π by itself.
3. **Teacher-directed**: A teacher who knows the target relation and learner's former predictions and wants to minimize the learner's mistakes by choosing π.
4. **Adversary-directed**: An adversary who knows the learner's algorithm and has unlimited computation power. His purpose is to maximize the mistake the learner makes.

3.1 General bounds

In director agnostic setting, we won't focus on the purpose of the query choice. We only consider the bound that applied to all kind of query sequences.

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

Proof. We can prove the bound by showing the algorithm. Assuming we have the adversary to determine the sequence. As the result of prior knowledge of learner's algorithm and unlimited computation power, the adversary can rearrange the matrix at its wish. For entries in first p columns, the adversary will tell the learner that its prediction is incorrect.

Also, in the entries of first q rows, the adversary will tell the learner that its prediction is incorrect. Now we consider the constraint on this p and q. Due to the limit of row types, the p and q cannot be too large. Since making mistakes in first p columns will reproduce at most 2^p row types, and mistakes in first q rows will make at least q row types, so we have

$$2^p + q \leq k$$

We set $2^p = \beta k, q = (1-\beta)k$, we can get the mistake lower bound

$$M \geq n\lfloor \log \beta k \rfloor + (1-\beta)km - n\lfloor \log \beta k \rfloor (1-\beta)k$$

□

Also the figure 2 illustrate this equation in a more intuitive way.

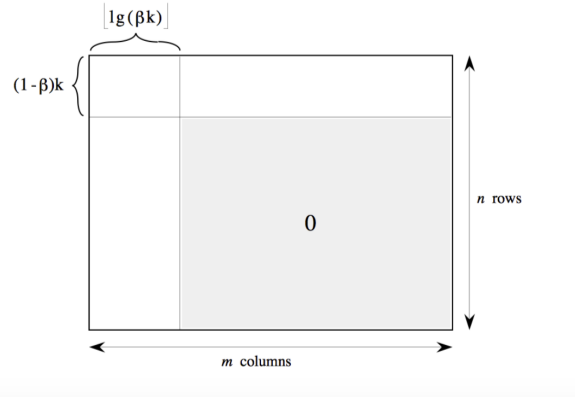

Figure 2: Illustration for lower query sequence in matrix form

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

Proof. This is trivial. There are no more than 2^{km} ways to select k row types. And also no more than k^{n-k} ways to assign each row to a certain type. So halving algorithm will make at most M mistake, where

$$M = \log(2^{km} \cdot k^{n-k}) = km + (n-k)\log k$$

□

3.2 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. Here's what algorithm do:

- Guess all entries for the first row. Record it as a row type.
- For the rest rows, predict the row i, column j's value by the majority vote of the recorded row type that consistent with the row i's known entries.

  If no such row type exists, query all the rest and record it as a new row type.

For each row type, we made at most m mistakes. For each of row has its type known before, we make at most $\lfloor \log k \rfloor$ mistakes. So combine the two parts we have the result. □

3.3 Teacher-directed Learning

Note that in teacher-directed setting, we only consider the upper bound making by the worst student it can have. That is because as long as we only consider the mistake bound, the smartest student can always have some sort of protocol with teacher that will reduce the mistake to some constant.

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. The teacher will first present 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. □

The reason why this bound is greater than self-directed learning is coming from the fact that we are facing the "dumbest" student who will use only minority vote to make the consistent prediction each time. Meanwhile, the log term in self-directed comes from the majority vote which can discard at least half of consistent row types by any mistake.

3.4 Adversary-directed

In adversary-director setting, the lower bound is the same as the general lower bound. So we only consider the upper bound. We may start from set $k = 2$.

Theorem 5. ***(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. Considering 2-colorable problem, we have a matrix and a graph at the same time. For each trial t, when the learner is querying M_{ij}, there are several possibilities:

1. no entry in column j is known, guess randomly;

2. all entries in j are the same. Predict the same;

3. Predict according to the row i' that is the same color as i.

Every time the learner make a prediction, other than updating the entry in the matrix, we also color the new vertex with a proper color. Also, connect it to some point that has a different color(if exist). □

When $k \geq 3$, this problem becomes NP-Complete, since it can be reduced to a k-colorable problem. We will set the problem aside here.

Bibliographic notes

The whole idea of learning binary relation comes from [1].

References

[1] Sally A Goldman, Ronald L Rivest, and Robert E Schapire. Learning binary relations and total orders. *SIAM Journal on Computing*, 22(5):1006–1034, 1993.