Advanced Concepts in Machine Learning
Assignment 3

Eliott Simon, Kevin Müller

November 2021

1 No. 5 - Tour de France

To predict which cyclist is most likely able to win the Tour de France, we need to choose a relational representation of the given data. Since the cyclists are organized in Teams, we cannot just measure the individual skill of each cyclist. The best individual cyclists depend heavily on the support they can get from their team mates, so in addition to measuring individual skill, the individual skill from a team mate can help to boost the individual skill from the teams' best cyclists.

After researching on how the Teams are structured, we read that some teams decide on one yellow jersey contender, whereas this cyclist is chosen to get the full support from every other team member and that particular cyclist is pushed to get the best individual result in the end for the team[1]. Other teams decide on more than one yellow jersey contender, so a computation for every yellow jersey contender per team is also not suitable, when there are teams that change their tactics during the tour.

Teams persists out of different cyclist types, where different cyclists have different specializations, such as sprinters, time trial specialists, mountain specialists or puncheurs, which are good all-around riders. Helper-riders are called domestiques, which are designated to gather water from team cars or ride in front of the Team to cut the wind for the other team members.

So a yellow jersey contender has to be good on an individual level, but also needs good specialists around him who can support him, by leading him to good individual times. For that reason a relational structure is needed, because individual skill from team members influence the skill from yellow jersey contenders.

The idea to compute individual skills depending on other team members is to create results from the past of each individual cyclist to be able to say where each cyclist has its strenghts. From past results every cyclist gets an individual score, so for the prediction of the new Tour de France, every cyclist gets a support

[1] https://www.sbnation.com/cycling/2018/7/17/17570042/tour-de-france-team-tactics-staff-support

score from each team member added to the individual score. If we would know how each stage in the Tour de France is structured (sprints, mountain stages), we could match the individual skills of the team to the structure of the Tour to predict which team and Individuals would be most likely to perform best, given their skillset.

- **Cyclist** - (c1 (cyclist identifier), name, skills mountain, skills sprint, skills overall, Team1, skills final)
- **hasResults** - (c1, t1 (tour identifier), s1 (stage identifier), final position, Team1 (team identifier), points sprint, points mountain, points overall)
- **hasTeammates** - (c1, Team1, c2 (cyclist identifier of team mate), skills mountain c2, skills sprint c2, skills overall c2)
- **Tour** - (t1, year, s1)
- **hasStage** - (s1, t1, score mountain, score sprint, overall score)
- **Team** - (Team1, team name)
- **hasTeammember** - (Team1, c1, skills mountain, skills sprint, skills overall)

Each individual cyclist has skills computed from his past results in "hasResults". These skills are dependant on the scores of "hasStage", because each stage in a race has different difficulty levels over the course. So when a stage would be sprint heavy, a good result would lead to increasing a cyclists sprint skills. The overall skills of a cyclist should be a combination of an individuals past results and a score factor that is dependant on the skills of his team mates for the upcoming Tour de France.

Possible Hypothesis:

- TourDeFranceWinner(c1 — :-cyclist(c1,name,...), hasResults(c1, t1,...), hasTeammates(c1, Team1, c2,...))

So the idea for the hypothesis is that we contain every result for every cyclist until a specific point in the past. We also want to list all the team mates of every cyclist, because we need their skills to compute the skills final of each cyclist. The cyclist with the highest skills final should then be the prediction for the winner of the upcoming Tour de France.

1.1 Problems

If we would know the difficulty levels of the whole upcoming Tour de France, we would also be able to compute how good all teams' cyclists match on the given course. If a team consists only out of good mountain cyclists and the tour is rather sprint heavy, that should be factored in in the computation of skills overall. So a problem is if the routes differ from year to year, we will not have sufficient records to learn with for the specific tour of a new year.

Another important question is how many times do cyclists change their team? There is also then a relation between past performances and the team mates an individual had at the given time. So if we look too far in the past, we may find data where a cyclist that now is a yellow jersey contender, was a support driver for someone else, which would influence the results.

1.2 Changing to Multi-Instance

A possible multi-instance representation could consist out of tables, where every cyclist would have row-entries for all stages he has attended and has results in. Individual skill scores could also be computed, but it would not be trivial to get the skill score from an individual that depends on the relation to other team members.

When we would not factor in the relation to other team members, we would lose the ability to compute the influence of support team members into the individual skill of a yellow jersey contender. This representation could only suite solo races like time trials, where time is measured for each cyclist individually, without driving together as a team.

2 No. 7 - Predicting ACML Grades

Predicting the grade for the course ACML is a hard task since this grade can depend on many factors. It is obvious that the grades previously obtained by the student (for other courses) can give insights on the grade they can achieve in ACML. However, most of the students had different education (come from distinct universities and had different bachelor).

2.1 Background

It is important to notice that the education is also a factor that can play a role in the student's grade for ACML. For example, a student with a Quantum Computing degree (imagine that this is a possible bachelor), will eventually do better at the quantum machine learning part of the exam, than other students. Since we have to deal with several databases, we have to use a **relational representation** for this task.
Henceforth, we can construct the following instance tables:

Name	**Prior Degree**	**GPA**
Tom	BSc Data Science	8.9
Sarah	BSc Electrical Engineering	7.0
David	BSc Computer Science	7.5
Jenifer	BSc Computer Science	8.5
Jenifer	MSc Quantum Information	7.8

It is important to notice that a student can have multiple degree, henceforth have two entries corresponding in this entry. This means that we can directly omit Attribute value (must be at least MI). For now let's consider that we could use multi-join to map the prior degree with the course (this would be valid). Note that it is not a good idea to put the prior degree in the table below, since if the student took 30 courses, this would just add the same degree 30 times in the entries. Instead, we can just add a relation between the course and the degree.

Name	**Course**	**Grade**
Sarah	Computer Security	5.0
Sarah	Data Analytics	8.0
Sarah	Data Mining	7.0
Tom	Databases	6.5
Tom	Discrete Mathematics	9.0
David	CS1	8.5
David	CS2	7.0
Jenifer	Quantum Machine Learning	8.4

*Note that this table is **m:n**, since a student can take different courses, and each course is taken by multiple students.

2.2 Resit?

Another important factor is to check whether a student took the resit, and the grade the student got for the resit. Essentially, a student can pass a course in the first sit, or in the second sit, AND s/he can fail the resit. Note that passing the resit implies that the student failed the first sit. We can model this as follows:

Name	**Course**	**isResit**	**Grade**
Sarah	Computer Security	False	4.0
Sarah	Computer Security	True	6.0
Sarah	Data Analytics	False	8.0
Sarah	Data Mining	False	3.0
Sarah	Data Mining	True	5.5
Tom	Databases	False	6.5
Tom	Discrete Mathematics	False	5.0
Tom	Discrete Mathematics	True	7.5
David	CS1	False	8.5
David	CS2	False	7.0
Jenifer	Quantum Machine Learning	False	1.0
Jenifer	Quantum Machine Learning	True	8.4

Although this would fit a multi-join (or tuple) representation, this not a good representation. The value that corresponds to the *isResit* depends on the previous grade (if grade ≤ 5.5, the next instance will always have a *isResit* that is True). If the student needs to retake a course many times (e.g pass on the fourth sit), this table could get very large.

Name	Course	Sit	Grade
Sarah	Computer Security	1	4.0
Sarah	Computer Security	2	6.0
Sarah	Data Analytics	1	8.0
Tom	Data Mining	1	3.0
Tom	Data Mining	2	5.5
Tom	Databases	1	6.5
Tom	Discrete Mathematics	1	5.0
Tom	Discrete Mathematics	2	7.5
David	CS1	1	8.5
David	CS2	1	7.0
Jenifer	Quantum Machine Learning	1	1.0
Jenifer	Quantum Machine Learning	2	8.4

This is essentially the same table as before, but the sit attribute became continuous, which becomes handy if the student has to retake the course many times.

Course	Degree
Data Mining	BSc Electrical Engineering
Computer Security	BSc Electrical Engineering
Discrete Mathematics	BSc Data Science
Quantum Machine Learning	MSc Quantum Information

The generalized hypothesis would look as follows:

```
grade(val) :-  course_grade( name , course(course_name, degree_name), sit , grade),
               degree(name , degree_name , GPA)
```

Essentially, this means that the grade depends on the grade of a given combination of course, degree, the i-th sit (if 1, it is the first sit), and the grade of the given course. It also depends on the degree that the student had, as well as the GPA obtained.
A concrete example of hypothesis would be:

```
grade(fail) :- course_grade( _ , course(Quantum Machine Learning, _ ), 1 ,  < 4.0 ),
               course_grade( _ , course(Quantum Machine Learning, _ ), 2 ,  < 6.0 ),
               degree( _ , _ , <6.0)
```

Where fail means that the student has a grade $\leq$5.5 at the ACML exam. This

means that the student (no matter what name) got a grade of at most 4 in the course 'Quantum Machine Learning' in the first sit, in any degree. Secondly, the student got a grade of at most 6 in the resit (same pattern). Finally, the student had a GPA that is lower than 6, no matter for which degree.
An even less generalized hypothesis could be :

```
grade(6.5) :-  course_grade( _ , course(Quantum Machine Learning, _ ), 1, [5.0-6.0] ),
               course_grade( _ , course(Data Mining, Data Science), _ ,  [7.0 - 7.5] )
               degree( _ , Data Science , [6.0 - 7.0])
```

Where the student will get a grade of 6.5 at the ACML exam, if s/he had a grade in range [5-6] at the first sit of the Quantum Machine Learning exam, and got a grade in range [7-7.5] in ANY sit of the data mining course in the Data Science Bachelor's. Besides, the student got a GPA that ranges between 6 and 7 at the BSc of Data Science.

3 Lower Language Representation

First of all, it is obvious that we can omit using an AV representation since we have this m:n relation between the student and the course. If we used an AV, we could not have a single row that defines all the grades of the student for all the courses, since a student can take many courses, and a course can be taken by many students. For this, we need **at least** a multi-instance representation, which allows use to define an instance by a set of row (e.g the grades of a student is a set of rows).
To depict this, we would have a table which looks as follows:

Name	**CS**	**DA**	**DM**	**DB**	**DM**	**CS1**	**CS2**	**QML**
Tom	Null	7	Null	Null	6	5	Null	5
Sarah	9	7	Null	Null	Null	5	Null	Null
David	Null	Null	6	Null	6	Null	9	Null
Jenifer	7	Null	5	Null	Null	5	8	Null

Note that these values do not match the previous tables, but this is just to show that AV would be bad at this task.
We see that there is a lot of Null values, where the student did not take the given course, hence the table is not very effective. Besides, this table does not represent the resit , and we would have to add a column to represent the resit of a given course, which would yield a very big table.
Secondly, multi-instance would not work when defining the course/degree relation. in the hypothesis, we have this nested relation course_grade((course, degree)), which could not be done using multi-instance representation. Using

different table and linking them through relational representations is the most effective way, and it prevents the MI table to lead to an explosion of information.