Capita AI, module 4.
Part B: Integrating learning and scheduling for an energy-aware scheduling problem

Tias Guns, *tias.guns@cs.kuleuven.be*

May 9, 2016

This module of the Capita AI course consists of two parts: the first, small, part consists of modeling a scheduling problem where the goal is to assign jobs to servers to minimize the total energy cost; the second part is the main part: to develop a framework that forecasts expected energy prices and solves the energy-aware scheduling problem such that the *real* energy cost would have been minimal.

Deadlines

- **New and optional: Sunday 15/05/2016 latest**. You can only learn by trying, so, if you send in a draft of your 4 page scientific report up to this date, I will give you some short feedback on your writing and reporting style.
- **Friday 20/05/2016 at 20:00** the latest: a scientific report of at most 4 pages with sections: 1) abstract summary of your findings (1-2 paragraphs), 2) method (overview of scheduling approach, learning approach and their integration), 3) results and discussion (of interesting experiments you did), 4) conclusions and future work (if you had more time, you would investigate...); as well as a zipfile of all your code, data and scripts. You can add an additional 1 page appendix with extra figures/tables.

The assignment should be made in the same teams as the previous assignments.

1 ICON energy challenge

As you know from Part A, the problem is based on the ICON Energy Challenge:
`http://iconchallenge.insight-centre.org/challenge-energy`

Here is once more the full description of the original challenge:

> "You are running a cloud computing service, where customers contract to run computing services (tasks). Each task has a duration, an earliest start and latest end, and resource requirements for CPU, Memory and I/O attributes. The tasks can be scheduled on one of multiple servers, each server has a limited capacity for the CPU, memory and I/O attributes. Multiple tasks can run concurrently on the same machine if the total resource consumption for all attributes is below the respective capacity. All tasks must be scheduled within their release and due dates, these dates are set so that no task stretches over midnight between two days. Tasks can not be interrupted, once started, they must run for their given duration. If a machine is used by a task, it must be running at that time. In addition to the cost of running the allocated tasks, the machine consumes some idle power if it is on. Every time a machine is switched on or off, a start-up resp. shut-down cost must be paid. All machines

are off at the beginning of the planning period, all machines must be off at the end of the planning period.

The price of electricity for the data centre is a real-time price, and varies throughout the day. The actual price is not known in advance, a forecast must be used to generate a schedule. The total cost of the schedule is determined after the fact by applying the actual price of electricity to the energy consumption in each time period. One forecast of the price is given by the organizers. However there may be a large discrepancy between the forecast and actual price, illustrated in the figure, offering the opportunity to generate better forecasts based on historical data for demand and prices, and previous forecast information. Note that a forecast with a low error is not automatically guaranteed to lead to a schedule with a low overall cost.

You use the provided price forecast to generate your best possible schedule. All tasks must be scheduled, the cost of the schedule is given by the actual price for electricity. The validity and cost of the schedule generated is tested by a solution checker. You generate your own price forecast, and use a provided scheduling system to generate a schedule. The cost of the solution will depend on the quality of your price forecast. You generate both the price forecast and a scheduling system. You provide a schedule, its cost will be determined by a solution checker."

In the challenge, the scheduling problem to solve was really hard. This forced all participants to focus only on the scheduling part, instead of the more novel interactions between the learning and the scheduling.

For this module of Capita AI, we will redo the ICON Energy Challenge, but we will use simplified scheduling instances so that the scheduling part is a lot easier and focus can be put on the interaction.

Part A is about writing a model of the scheduling problem in the MiniZinc language. **Part B** will be about developing a learning method and experimenting with different approaches.

I provide (simplified) scheduling instances of different hardness, and the historic data of the ICON challenge. I also provide some basic python scripts that can help you start faster.

2 Part B: integrated modeling and scheduling

1. The scheduling In Part A, you developed a model of the scheduling problem in MiniZinc. Be sure to keep the 'output' statement intact, otherwise the scripts wont be able to parse the output.

Each `loadX/` directory contains 14 scenario's named day01.txt .. day14.txt. These represent a scenario of loads for 14 consecutive days on a small cluster. Apart from being for consecutive days, the scenario's are not not explicitly tied to a particular date. Choosing a starting date for a load (and comparing the effect of different starting dates throughout the year) is part of your evaluation.

2. Energy price data In the `data/` directory you will find the file `prices2013.dat`. This is a comma-separted value (CSV) file of historic data from 1/11/2011 till 31/12/2013. See the ICON Energy challenge description PDF for an explanation.

Note that like all real data, this is **dirty** data. For example:

- The feature names in the PDF do not literally match the ones of the datafile ('ForecastWind-Production' = 'WindForecast', 'SystemLoadEA' = 'LoadForecast', 'SMPEA' = 'PriceForecast', 'SystemLoadEP2' = 'ActualLoad', 'SMPEP2' = 'ActualPrice', ...)
- There is **missing** data. This is indicated by a 'NaN' entry in the data. Especially the features 'ORKTemperature' and 'ORKWindspeed' contain many missing values. If you wish to use those features, you should deal with them properly, e.g. by imputing values.
- Negative energy prices are not noise, prices can actually be negative on the real market (e.g. when network almost overloaded by renewable+nuclear energy)

- ... there may be other dirty/noisy things

Furthermore, we assume we are operating in a 'day-ahead' scheme (the official challenge sometimes speaks of a 4-day delay, we assume a 1-day delay). This means that at prediction time (and hence learning also), you may not use the features 'ActualWindProduction', 'SystemLoadEP2' and obviously 'SMPEP2' of the entire day for which you are predicting the prices. These features are not known when doing the day-ahead predictions.

3. Learning and forecasting of the energy price You can implement any aspect of this task in any programming language that you want. However, to help you get started quickly I have created a few **example python scripts**. The file `prices_data.py` contains routines for easily reading and querying the energy price data. The file `prices_regress.py` contains some sample code that uses this data and the *scikitlearn* machine learning toolbox [1], to learn a linear regression and SVM regression model on 30 historic days to predict for the week ahead. I also added some code that visualizes the real and predicted prices, to encourage you to visualize your data during development too.

Run `prices_regress.py` (from the `scripts/` directory) to see what it does, and check the source for details...

(the `prices_regress.py` script is simple and does not deal with missing values, hence if you get the error 'NameError: name NaN is not defined' it randomly picked a period with a missing value)

4. Integrating learning, forecasting and scheduling I have generated 8 possible scheduling scenarios `load1/..load8/`. Each contains 14 files that simulate a workload of 14 consecutive days on a cluster. As written before, it is up to you to choose the date(s) that you want to link to a load during evaluation.

However, you should start from a starting date, where you learn from historic data (e.g. a number of previous days/weeks) and predict the next day, then schedule day01.txt using the predicted prices. Then you (possibly after relearning) predict the day after, and schedule day02.txt, etc until all 14 days are scheduled.

There are still a large number of design choices open, both for the algorithm and the evaluation. You are free to implement an approach in any programming language that you wish. To get you started faster I've implemented a rather straightforward prototype in the file `mzn-prototype.py`. It has a few parameter options; will load the data and learn one model based on the past 30 days; it will make predictions for the 14 days following the (random) first date; and run the scheduling problems using these predictions.

An example run is as follows:

```
./mzn-prototype.py energy-teamtias.mzn load1
First day: 2013-01-08
Plotting actuals vs predictions...
Residual sum of squares:
me: 1096.03
scheduling_scenario; date; cost_forecast; cost_actual; runtime
load1/day01.txt; 2013-01-08; 2049077.3754; 1966343.5023; 0.66
load1/day02.txt; 2013-01-09; 1654869.6521; 1452898.2245; 0.72
load1/day03.txt; 2013-01-10; 1636601.0627; 1667239.2343; 0.48
load1/day04.txt; 2013-01-11; 2206826.2711; 2291498.9873; 0.55
load1/day05.txt; 2013-01-12; 1015339.2463; 933508.9834; 0.56
load1/day06.txt; 2013-01-13; 1691418.3372; 1778219.4181; 0.65
load1/day07.txt; 2013-01-14; 1735863.3579; 1743641.6610; 0.52
load1/day08.txt; 2013-01-15; 1676947.6280; 2025041.6795; 0.54
load1/day09.txt; 2013-01-16; 1246022.0094; 1213653.1933; 0.49
```

[1] SKLearn has a number of dependencies. For windows users, Lieven Paulissen reported succesfully installing the python wheels using the pip command, especially numpy and matplotlib, from here: `http://www.lfd.uci.edu/~gohlke/pythonlibs`

```
load1/day10.txt; 2013-01-17; 2129447.6472; 2002380.1243; 0.65
load1/day11.txt; 2013-01-18; 2015314.1242; 2088155.0670; 0.61
load1/day12.txt; 2013-01-19; 1657882.3124; 1597138.8799; 0.54
load1/day13.txt; 2013-01-20; 1346852.7618; 1464254.4839; 0.48
load1/day14.txt; 2013-01-21; 2054393.0086; 2096042.0586; 0.61
load1 from 2013-01-08, linear: Total time: 8.06 -- total actual cost: 24320015.5
```

Your goal is to do better (and/or more interesting) things than what this script does.

Possible directions

The linear regression + scheduling approach of the `mzn-prototype.py` script is pretty straightforward. The challenge for you is to do better. Part of the challenge is also to define an evaluation method that defines when something is better, though it must involve the actual cost of the schedules in one scenario.

Apart from the evaluation, there are many many things you can try to obtain better results. Some ideas should you need inspiration:

Preprocessing

- How much data is used?
- Do you deal with missing values, if so how?
- What features do you use? How did you select them?
- Do you transform your features? Do you create additional features?
- Dealing with peak prices: instead of a regression problem, discretize the price and treat as classification?
- Do you learn one model or group the data in some way and learn separate models? e.g. per week-day, or per period-of-day, or ...
- Change the data to be temporal and include features of previous timesteps/days for a single prediction? (careful: no SMPEP2 of same day timesteps!)
- What about clustering the data to extract groups?
- Other analysis techniques to better understand the data?
- ...

Learning method

- Different regression methods?
- Predict price class rather than regression? how to define classes?
- Instead of predicting the expected value, use confidence intervals? Or quantile regression?
- Cost-senstive learning? what cost? (extract from instance, e.g. min/max usage?)
- Ensemble methods?
- Parameter tuning? Cross-validation?
- ...

Integration

- For quantile/confidence: use a non-0.5 quantile or take both lower and upperbounds into account in the scheduling?
- Make the learning aware of the scheduling tasks (cost-sensitive or otherwise)?
- Iterate between forecasting and scheduling within same day? (can not use actuals for this!)
- One model for all days, or retrain every day? Or something else?
- ...

These are just ideas, you can try anything you want, as long as you have interesting results to share at the end of the ride (e.g. through comparisons).

Professor De Raedt indicated that the expected time to be spent on the assignment is about 20-25 hours.

Evaluation

You will be evaluated on 3 criteria, each quoted on 6 points:

- **Depth** How non-trivial and in-depth is your integrated approach? What kind of preprocessing and learning method(s) do you use, what setup do you use for the different days and timeslots, does the scheduling problem change?
- **Quality of the integrated method** Is the evaluation scientifically sound (multiple loads, multiple days, ...)? How does your method compare to other methods that you tried (and perhaps to my prototype?)
- **Quality of the scientific report** Is the writing style appropriate? Are the descriptions and arguments clear? Is there enough yet not too much detail? Are the results fairly presented?

You can additionally win or loose a point based on the creativity and novelty of your solution: if it is potentially publishable you get +2, if it makes no effort to go beyond the prototype or what is in the Ifrim et al. paper you get +0, otherwise you get +1.

Report format

The report should be in 10pt font, a4 papersize, with standard 'wide' margins.

In latex, this is obtained with (see this document):

```
\documentclass[a4,10pt]{article}

\usepackage[a4paper,includeheadfoot,margin=2.54cm]{geometry} % was \usepackage{a4wide}
```