IN-STK-5000, Medical Project

Christos Dimitrakakis `chridim@ifi.no`

December 12, 2019

Before you start, make sure you have

- Joined one of the project groups in Piazza.
- Forked the code in `https://github.com/olethrosdc/ml-society-science`
- All questions should go through the QA platform in Piazza.

1 Historical data (Nov 6)

First, we shall take a look at historical data present in Matlab/Octave format in `data/medical/historical.dat` For each patient, we observe the attributes $\boldsymbol{x}$, with

- $x_1 \in \{0, 1\}$, sex.
- $x_2 \in \{0, 1\}$, smoker.
- $x_{3:128} \in \{0, 1\}^{125}$, gene expression data. These variables can have missing values, but here they are all included in the historical data.
- $x_{129:130} \in \{0, 1\}^2$, symptoms. These can be taken to be akin to labels in supervised learning. There may be missing data here too, but for now we can assume they are included.

We also observe a therapeutic intervention $a \in \mathcal{A}$, which is followed by an outcome $y_t \in \{0, 1\}$. Consequently, historical data can be described by $(\boldsymbol{x}_t, a_t, y_t)$

Discovering structure in the data. It is uncertain if the symptoms present are all due to the same disease, or if they are different conditions with similar symptoms. (a) looking at only the attributes, estimate whether a single-cluster model is more likely than a multiple-cluster model. You can use anything, starting from a simple clustering algorithm like k-means to a hierarchical Bayesian model. (b) Try and determine whether some particular factors are important for disease epidemiology and may require further investigations.

You need to be able to validate your findings either through a holdout-set methodology, appropriately used statistical tests, or Bayesian model comparison.

Measuring the effect of actions. We also observe the effects of two different therapeutic interventions, one of which is placebo, and the other is an experimental drug. Try and measure the effectiveness of the placebo versus the active treatment. Are there perhaps cases where the active treatment is never effective, or should it always be recommended?

2 Improved policies (Nov 20)

The data we have observed comes from some policy π_0, taking actions in $\{0, 1\}$. Now we have to be more specific about what the meaning of each treatment and outcome variable is. In this case, $a_t = 0$ is a placebo, and $a_t = 1$ is an experimental drug, with $y_t = 0$ meaning no effect, and $y_t = 1$ meaning a measurable effect. For the purposes of this exercise, you can assume the utility function is:

$$U = \sum_t r_t, \tag{2.1}$$

$$r_t \triangleq -0.1a_t + y_t. \tag{2.2}$$

The -0.1 factor implies that the active treatment must be at least 10% more effective than the placebo.

Use the skeleton `random_recommender.py` to implement your code.

Assumption 2.1. *The observed policy π_0 and all other policies π can be represented as conditional distributions $\pi(a \mid x)$.*

Exercise 1 (Measuring utility). In this exercise, you should implement the `estimate_utility()` method to estimate the utility of policies on the historical data `(data, actions, outcome)` passed to the function.

1. Measure the utility of the historical policy π_0 on the historical data.
2. Provide error bounds on the expected utility and explain how those were obtained.

Exercise 2 (Improved policies).

1. Find an improved policy $\hat{\pi}$, taking actions in $\{0, 1\}$. *Hint: This can be done by simply selecting, for each x_t, the action a_t maximising expected reward according to your model, as the utility is a sum of rewards. Of course, you should first build a model.*
2. Calculate the expected utility of the improved policy $\hat{\pi}$ from the historical data. Implement this in `estimate_utility()`

3 Adaptive experiment design (Dec 6)

For this part of the exercise, make sure that you have implemented everything within the API defined in the `random_recommender.py` skeleton. All the methods you have implemented for feature selection, dimensionality reduction, policy evaluation, etc. must be self-contained within the recommender. The test bench will call your recommender's `fit_data()` and `fit_treatment_outcome()` with possibly new datasets. This will be done mainly in order to test how your algorithm behaves with varying amounts of data.

Exercise 3 (Online policy testing). Here, make sure that you recommenders are based on `random_recommender.py`. You should be able to run all your experiments on `TestRecommender.py`

1. Create a `HistoricalRecommender` recommender class that estimates the historical policy π_0 when `fit_treatment_outcome()` is called. Measure the expected utility of the original policy π_0 using the Test Bench.
2. Create a `ImprovedRecommender` class that implements your creates an improved policy $\hat{\pi}$ when `fit_treatment_outcome()` is called, and uses this policy when `recommend()` is invoked. Measure the expected utility of your improved policy $\hat{\pi}$ using the Test Bench `TestRecommender.py`
3. How do those differ from the results you obtained on historical data?

Exercise 4 (Adaptive experiments). In this setting you should contrast your `ImprovedRecommender` with the `HistoricalRecommender` and an `AdaptiveRecommender`, that you should implement, which should change its recommendations as it obtains new data through `observe()`. We will consider two cases:

1. The same set of treatments are available.
2. There is an additional set of experimental treatments. For the additional treatments, make sure that line 36 of the Test Bench to loads a bigger action matrix:

   ```
   generator = data_generation.DataGenerator(matrices="./big_generating_matrices.mat")
   ```

   2 is a potentially improved treatment for the general population, while the remaining target specific genes. Hence, you might expect these to be effective against only a small portion of the patients.

In either case, you can have two alternative goals: (a) discover the most effective treatment policy at the end of the trial (b) maximise the expected number of people to be treated.

For simplicity, use goal (b) to plan the experiments. However, at the end, you should report which treatment policy appears to be the best.

Since the data will be dynamically generated, implement the final analysis in your function. This can show

1. Recommending a specific fixed treatment policy
2. Suggesting looking at specific genes more closely
3. Showing whether or not the new treatment might be better than the old, and by how much.
4. Outputting an estimate of the advantage of gene-targeting treatments versus the best fixed treatment