Advanced Topics in Control 2020: Large-Scale Convex Optimization

Exercise 10: Classification

Goran Banjac, Mathias Hudoba de Badyn, Andrea Iannelli,
Angeliki Kamoutsi, Ilnura Usmanova

May 19, 2020

Date due: May 28, 2020 at 23:59.

Please submit your written solutions via Moodle as a PDF with filename `Ex06_Surname.pdf`, replacing `Surname` with your surname. Include any code as separate files (not inside the PDF itself), and please don't zip anything.

---

1 Problem 1

You have decided to dedicate yourself to a life of organized crime. After battling your way across Nugget Bridge, you are approached by a grunt who offers you membership in the legendary crime syndicate Team Rocket. Unfortunately, the grunt notices that your skills at battling Pokémon are not as good as your skills with optimization, and so you are relegated to their research and development department.
Unable to steal a Pokédex from Professor Oak, Team Rocket wants you to develop an alternate method for studying Pokémon based on observing them in nature. A Pokémon can have one or two of 18 types:

Normal, Fire, Fighting, Water, Flying, Grasss, Poison, Electric, Ground,
Psychic, Rock, Ice, Bug, Dragon, Ghost, Dark, Steel, and Fairy.

As a Pokémon grows in level, it learns various moves that it uses in apparently perfectly legally sanctioned and obviously not cruel at all battles against other Pokémon at the order of their human trainers. You have a fantastic idea of developing a classifier that will be able to tell the type of a Pokémon based off of the moves it uses in battle.
Note that the computation time for this problem set can be substantial. Be sure to start this one early, so you have time to set it aside for a few hours to run.

(a) Jesse and James have stolen a bunch of data from the reigning Pokémon champion, but the data is much more extensive than we need. In this problem, you must use your favourite programming language `Python` to curate the data into more useable data structures.

In `pokemon.csv`, you have a table that contains, among other things, all 801 Pokémon and their types. Convert this data into a matrix with 801 rows, corresponding to the Pokémon's

Pokédex ID, and 18 columns, corresponding to their type. The matrix B should be of the form

$$B_{ij} = \begin{cases} 1 & \text{Pokémon } i \text{ has type } j \\ 0 & \text{Pokémon } i \text{ doesn't have type } j. \end{cases} \tag{1}$$

For example, Charizard has Pokédex ID 6, and has the Fire and Flying types. Therefore, $B_{6,j} = B_{6,k} = 1$ where j and k are the columns corresponding to Fire and Flying. Of course, `Python` starts indexing at 0, so keep that in mind.

(b) In this next part, you will curate the data structure corresponding to the Pokémon moves that it learns as levels up, or that can be taught to the Pokémon by use of a technical or hidden machine. In `pokemon_moves.csv`, you will find a dataset containing, among other things, a column corresponding to the Pokémon ID, and a column corresponding to the move ID number that the Pokémon can learn. There are a total of 728 moves, and so we want a data structure M whose columns correspond to Pokémon, and whose rows correspond to the moves they can learn. Create an array M such that

$$M_{ij} = \begin{cases} 1 & \text{Pokémon } j \text{ can learn move } i \\ 0 & \text{Pokémon } j \text{ can't learn move } i. \end{cases} \tag{2}$$

(c) I've actually already compiled the data for you: M is in `ex10_pokemon_moves.csv`, and B is in `ex10_pokemon_types.csv`. Feel free to use these datasets to make sure you did the previous two problems correctly. The rows of M correspond to the moves in `moves.csv`, and the columns of B correspond to the types in order from the paragraph above Part (a).

Now, using `cvxpy`, we will implement logistic regression to determine the Pokémon type from its moveset. In logistic regression, we are given pairs of data (x_i, y_i) where $x_i \in \mathbb{R}^n$ is a *feature vector*, and $y_i \in \{0, 1\}$ is a *boolean class*. We want to contruct a *classifier* $\hat{y}$

$$\hat{y}(x) = \begin{cases} 1 & \beta^T x > 0 \\ 0 & \beta^T x \leq 0. \end{cases} \tag{3}$$

that tells us if x is in the class or not. In our case, the feature vector x_i is the moveset corresponding to Pokémon i, and $y_i^{(k)}$ is the boolean telling us if Pokémon i is the type k or not. Hence,

$$M = \begin{bmatrix} | & & | \\ x_1 & \cdots & x_{801} \\ | & & | \end{bmatrix}, \; B = \begin{bmatrix} y_1^{\text{normal}} & y_1^{\text{fire}} & \cdots & y_1^{\text{fairy}} \\ y_2^{\text{normal}} & y_2^{\text{fire}} & \cdots & y_2^{\text{fairy}} \\ & \vdots & & \\ y_{801}^{\text{normal}} & y_{801}^{\text{fire}} & \cdots & y_{801}^{\text{fairy}} \end{bmatrix}. \tag{4}$$

To find the classifier for type (k), the objective function we want to maximize is

$$\mathcal{J}(\beta) = \sum_{i=1}^{n_{\text{training}}} y_i^{(k)} \beta^T x_i - \log\left(1 + \exp(\beta^T x_i)\right). \tag{5}$$

Implement this in `cvxpy` using the first 400 Pokémon for training, i.e. $n_{\text{training}} = 400$, for the normal type, i.e.

$$\beta^{\text{normal}} = \text{argmax}_\beta \sum_{i=1}^{n_{\text{training}}} y_i^{\text{normal}} \beta^T x_i - \log\left(1 + \exp(\beta^T x_i)\right). \tag{6}$$

Don't forget that `python` begins indexing at 0.

Does your classifier return that Snorlax is a normal-type Pokémon?

(d) Now, run the optimization problem in the previous part for all 18 types to obtain classifiers for each type:

$$\beta^{\text{normal}}, \cdots, \beta^{\text{fairy}}. \tag{7}$$

Again, use the first 400 Pokémon for your training set.

Then, test the classifers on the remaining Pokémon, i.e., those with Pokédex ID 401-801. For each type, return the following:

(a) True positives $T_+^{(k)}$: $\hat{y}^{(k)}(x_i) > 0$, and Pokémon i is type k

(b) True negatives $T_-^{(k)}$: $\hat{y}^{(k)}(x_i) \leq 0$, and Pokémon i is not type k

(c) False positives $F_+^{(k)}$: $\hat{y}^{(k)}(x_i) > 0$, but Pokémon i is not type k

(d) False negatives $F_-^{(k)}$: $\hat{y}^{(k)}(x_i) \leq 0$, but Pokémon i is type k

(e) For which Pokémon type is the classifier most accurate? For which one is it the least accurate? Why is this the case? **Hint:** This requires knowledge of the Pokémon games to solve. **Other Hint:** Think about what the types mean. Is there such a large difference between Grass and Poison types?

(f) Let's try and mess around with the training data to improve the classifier performance. We can measure the classifier performance with the following ratios:

$$r_1^{(k)} = \frac{F_+^{(k)}}{T_+^{(k)}}, \; r_2^{(k)} = \frac{F_-^{(k)}}{T_-^{(k)}}, \; r_3^{(k)} = \frac{F_+^{(k)} + F_-^{(k)}}{T_+^{(k)} + T_-^{(k)}}. \tag{8}$$

Try various values of n_{training}, say $100, 200, 300, 400, 500, 600$ and 700 (using the remainder of the Pokémon as testing data), and plot the values of the ratios against n_{training} for each type. Do these ratios make sense as metrics of performance? Discuss, and if necessary, modify the ratios or propose your own metric of performance and test your classifiers against it.

(g) Once you settle on a 'optimal' value of n_{training}, run your optimization algorithm again to obtain for all types k,

$$\beta^{(k)} = \text{argmax}_\beta \sum_{i=1}^{n_{\text{training}}} y_i^{(k)} \beta^T x_i - \log\left(1 + \exp(\beta^T x_i)\right). \tag{9}$$

This time, instead of testing each type separately, report the true/false positive/negative rates over all *single*-type Pokémon, and over all *dual*-type Pokémon. For the dual-type Pokémon, double-count each type, i.e. for Bulbasaur who is both Grass and Poison, test it both for Grass and Poison. Is the classifier more accurate for single- or dual-type Pokémon?

If your are interested, Jesse and James stole the data from:

- `https://github.com/veekun/pokedex/tree/master/pokedex/data/csv`
- `https://www.kaggle.com/rounakbanik/pokemon/data?select=pokemon.csv`