Pattern Recognition - Project 01
Predicting Battery Usage of a Phone

Tutor: Sebastian Mueller
Handed out: **Nov 23**
Discussion: **Dec 21**
Submission deadline: **Dec 21, 1615h**

The Dataset

The dataset 'android.csv' was recorded on an android phone and contains information regarding the battery level and what apps were running at a given point in time. The header contains the following labels:

- battery_level: change in battery since last measurement, negative values correspond to the battery discharging
- battery_plugged: whether battery was charging since last measurement
- battery_status: no idea
- day, month, year: day, month, year
- slot: refers to a 'timeslot' where the day was split into 30 minute intervals
- packages_running_*: binary indicator whether app "*" is running

Task

The goal is to design a system that predicts the battery usage between two points in time. The input is an "acitivity vector" indicating what apps are running and also includes information on battery_plugged.

The finished system will consist of two components:

1. will predict what states will occur in the next steps. For that you will use Markov Chains and approximate the parameters using
   (a) Maximum-Likelihood estimation
   (b) Maximum a posteriori parameter estimation, using a uniform prior for the transition matrix
2. will give an estimate for the battery usage given an activity vector. For this you will fit a linear regression in two different ways
   (a) MLE parameter estimation
   (b) MAP parameter estimation

Use the first component to generate a batch of states given an initital state and feed them into the regressor. Accumulate the output of the regressor to arrive at an estimate for battery usage.

What libraries/ functions from the standard python stack can you use? The general notion is that you should implement the *algorithms* yourself. This means that arithmetic operations provided by numpy, functions like exp(.) or log(.) and random samplers for different distributions from numpy or scipy are fine. Using functions or classes that implement the hole algorithm, e.g. LinearRegression in sklearn, is not allowed.

Timeline

We split this mini-project into 4 packages. In the first week you should work on the first package to get acquainted with the data. In weeks two and three you can focus on implementing and evaluating the different approaches. The tutorial sessions during that time will be used to have discussions on problems you encounter. The last week is for you to assemble everything you did and learned in a summarizing document/ presentation.

1 - Exploring the data

At first you should make sure that the data is "clean" and then look at some basic statistics to get a feeling for it. For example you could answer questions such as:

- Of what data type are the features?
- Are all battery level readings plausible?
- Are there fields that do not contain a value or are 'NaN'?
- How often is each app running?
- How often is the battery change positive/ negative?
- ...

Also see **Addendum: Visualizations** at the end of this document.

2 - Predicting Battery Usage

After taking a look at the data you can now start to work on the two component system. The state predicting component will evolve around Markov Chains and the battery change predicting component will be a regressor. Following "Occam's razor", the solution should be as simple as possible. This saves copmutational resources, makes debugging easier and increases maintainability. A practical (greedy) way to reach such a solution is done by first choosing a basic algorithm (such as ordinary least squares) and then expand on it based on interim results (such as bad performance). You should at least ipmlement and evaluate what is listed below, but if you have another idea you want to try out feel free to do so!

Some guiding questions to get you started:

State prediction

- The samples are high dimensional binary vectors: Is this a "state" as needed by the Markov Chain or do you need to do something with it?
- What states shuold be considered adjacent?
- Are all states reachable or "leavable"?

Regression

- Are there apps that you can exclude from the activity vector?
- Is it sensible to have the same regression model for the cases where the phone is charging and where it is discharging?
- Can you do something to the feature vector to give it more useful information?

3 - Training and Evaluation

For each of the two components you will try out different configurations. But how do you know which one to choose in the end?

k-fold cross-validation

General scheme K-Fold cross-validation "validates" your results in the sense in that it minimizes the risk that the training/ testing performance you observe is not an artifact of the data selection process but actually reflects the model behavior. You already know about splitting data into a training and a testing set to assess generalization capabilities of a model. However if you do just one random split it is possible that by sheer good or bad luck the model performance is better or worse than it would have been with any other split. To rule such random effects out one uses the k-fold cross-validation scheme:

1. Split dataset into k disjoint and equally sized subsets
2. Use all but one set for training and collect training performance
3. Evaluate on the set you did not train on and collect testing performance
4. Repeat until each of the k sets served as the testing set once
5. Compare results for all sets to arrive at a conclusion whether or not the model behaves robustly

How do you apply this procedure to validate your models?

State prediction The Markov Chain assumes that a state depends on its predecessor. However shuffling the data randomly and computing a stochastic matrix from this split might violate this assumption. What might be a reasonable way to split the dataset into subsets that does not fundamentally violate the underlying process?

Battery level Splitting the set randomly is not problematic here since we can assume that the energy consumption only depends on the current state and not the previous ones. This means we do not need to keep the samples in any specific order. The most extreme form of k-fold cross-validation is called leave-one-out cross-validation. Leave-one-out means that the testing set has one single sample. Thus you get as many training-testing splits as you have data samples. This split is especially useful if, as in our case, one only has access to a very limited amount of data.

4 - Presenting your results

You can present your results in the tutorial session on Dec 21. Additionally, the registered groups may also submit either a written report in the form of a PDF or a (executable) Jupyter Notebook. Your report should ...

- ... describe or show how you analyzed the data and describe what design choices this inspired
- ... constitute a full description of your system
  - how does the input look like?
  - what components does it have?
  - what further processing do the components do to the input before generating the output?
  - How do they produce the output? For example do you do some post processing?
- ... show how you analyzed the performance of the system
  - How accurate is the state prediction?
  - How accurate is the regression?
  - show how you validate your results

Make sure your plots at least are of an appropriate size, have a title and labeled axes; a short description would be even better.

Addendum: Visualizations

Numbers are great but if you need to look at too many of them at the same time, for example in the case of the leave-one-out cross-validation, visualizing them helps a lot. **Visualizations are not only great to explore your results but are also indispensible for data analysis**. Some ideas for visualizations:

- app usage: histogram
- k-fold cross-validation: box plots; multiple models on x-axis for better comparability
- state generation: line/ scatter plot of the distance between target and predicted states for multiple steps into the future.

- regression parameters: bar plot
- regression performance: line/ scatter plot for the battery level and the predicted battery level in the same coordinate system