Using Random Fourier Features with Random Forests

Deliverable 1: Context and scope of the project

Albert Ribes Marzá

March 6, 2018

Contents

1 Project Formulation **3**

2 Scope **3**

3 Possible obstacles and solutions **4**
3.1 Trouble in modifying the code 4
3.2 Computational power 4
3.3 Find suitable public datasets 4

4 Methodology **4**

5 Development tools **5**

6 Validation Methods **5**

1 Project Formulation

Random Forest is a very powerful ensemble learning method than can be used both for classification and regression. It is based on building a group of Decision Trees (a forest) and output the prediction of the majority of them, and it corrects the problem of Decision Trees of over-fitting on the data[2].

The performance of this method can be very high, and it is a very commonly chosen methods for the following reasons[1]:

- It has a hight execution speed
- It doesn't over-fit on the data by increasing the amount of trees in the forest
- It can handle huge amounts of data
- It can achieve very hight accuracy
- It generates an estimate of the generalization error

However, in order to generalize on unseen data, the correlation between any two trees in the forest need to be very low. For this reason, it is needed a randomized sampling method to train each tree in a different way. At the same time, each of the trees should have a low error rate, so having a good sampling method is important.

Many methods have been proposed, but most of them are based on hiding part of the data to the Decision Trees, and hence it affects on the performance of each individual one.

I propose a different approach to lower the correlation among the trees in the forest. Instead of feeding them with the features of the original dataset, I suggest to map them to a randomized low-dimensional feature space, allowing the trees to use different mappings on the same data. This transformation is based on Random Fourier Features and it has been verified that it approximates various radial basis kernels[5].

This approach has already been done with Deep Neural Networks[3] and has showed to be very effective, but it still haven't been studied on Random Forests. That's what I will study in this project.

2 Scope

The project will focus on training Random Forest with Random Fourier Features for classification problems.

It will test the algorithm proposed against currently used Random Forests methods to see if it outperforms on accuracy or train time.

To do so, the problem will first be addressed in the theoretical approach to find out the best way to mix those two techniques.

Then, I will implement the ideas found in the first section in a programming language. I will use Python 3, since it is a powerful yet flexible language, suitable to work on machine learning and maths. In addition, it will allow me to use the *scikit-learn* tools[4], which contain a module with a Random Forest Classifier. It is a well tested library, and as it uses the 3-Clause BSD License, I will use

the source code to build the Random Forest Classifier using Random Fourier Features, so I will not have to do it from scratch.

Finally, I will test my implementation using public classification problem datasets, together with currently used Random Forest algorithms, in order to find the best tunning parameters and to see if it outperforms the state-of-the-art methods.

3 Possible obstacles and solutions

3.1 Trouble in modifying the code

The plan is to modify the Random Forest Classifier implementation of the sklearn library to make it work as my algorithm describes. It is possible that it is so complicated to understand that I will not be able to modify it. If this is the case, I can use another equivalent implementation, for instance the one from R, which my project tutor happens to know, so I can receive help from him.

If the problem persists, I will have to implement the code from scratch.

3.2 Computational power

The kind of problems the algorithm is expected to tackle are the ones with huge amounts of data to train. Although the training from this data is feasible for research groups with high computational power, I will only be able to use a laptop for the training. It may happen that I can not afford to commit the required time for training the data.

If this happens, I will make the tests with tinier datasets and hope it to be representative to the real problems.

3.3 Find suitable public datasets

I plan to use public datasets to test my algorithm, but it may happen that I am not able to find a dataset with the required characteristics, such as type of problem, number of instances and attributes, etc.

If this happens, I can create my own dataset with dummy data, as the aim of the project is not tied to a specific domain.

4 Methodology

During the realization of the project I will use the Waterfall Project Management methodology. The reason to chose that methodology is that the scope of the project is very well defined, and most of the tasks to do need to follow a sequential order. It is a simple methodology and it fits the requirements of this project.

I will make a list of all the steps I need to accomplish the deliverable, assign a sequential sorting and decide when each particular task needs to be done. Then, I will start doing them one after the other.

If it is possible, some of the tasks may overlap in time, as it may be useful to better match the different parts of the project.

As the methodology is so simple, there are many tools and ways to follow it. The main tools I will use to follow it will be:

- **Trello**: a web based project management application to track the work to do, work in progress and work done.
- **Google Calendar**: To keep track of the deadlines and be notified on delivery days.
- **todo notes and todo notes trackers** such as todo-show from Atom text editor, to mark all the pending work.

5 Development tools

The realization of the project will require the following development tools:

- **Git and Github**: used as a revision control system of the code and to the reports and securely store it in the cloud.
- **Python Programming Language**: all the code will be writen in this language
- **Python modules and libraries**: used for advanced maths and machine learning features, such as the Random Forest Classifier class from scykit-learn.
- **Text editor**: to write and modify the Python code
- **Testing datasets**: to evaluate the algorithm
- **A laptop**: for obvious reasons

6 Validation Methods

As each task will have a pre-fixed deadline, the validation method will consist of checking at a given date if I have finished all the things that needed to be done.

Each week, at a definite day, I will check the list of things to do and the list of things done, and I will evaluate if the goals have been achieved.

References

[1] Leo Breiman and Adele Cutler. *Random Forests*. URL: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm.

[2] Tin Kam Ho. "Random Decision Forests". In: (1995).

[3] Siamak Mehrkanoon, Andreas Zell, and Johan A.K. Suykens. "Scalable Hybrid Deep Neural Kernel Networks". In: (2017).

[4] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[5] Ali Rahimi and Benjamin Recht. "Random Features for Large-Scale Kernel Machines". In: (2007).