ITN March-Retreat Project Report
Implementing Complex Mathematical Expressions using Python

Max Zwießele

March 18, 2015

Many mathematical models depend on optimizing an objective function with respect to its parameters, which have to be found from the data itself. In maths, maximizing a function with respect to its parameters usually involves going along the gradient space, which is spanned by the parameters. One can think of a ball, rolling down a valley, to eventually find the lake at the bottom (the optimial point of parameters). One problem of complex modeling and complex expressions is that the gradients for the parameters can be hard to derive, and therefore, a quick prototyping of new ideas is restricted.

In our project we combined two projects into one framework, which allows quick prototyping of gradient based optimization, without the need to explicitly write down the gradients of the objective function at hand, called *SymDiffPy* (`https://github.com/mzwiessele/SymDiffPy`).

We used GPy (`https://github.com/SheffieldML/GPy`) in order to handle parameters and optimization in order to give an easy to use handle on the parameters and objective function. GPy was originally designed for parameter optimization of the so called Gaussian process, which is a probabilistic objective function. Its efficient parameter handling and easy to use object oriented implementation makes it the perfect candidate for a project of function optimizing.

For the second step of computing the gradient of the objective functions at hand, we used the so-called package Theano (`http://deeplearning.net/software/theano/`), which is able to automatically compute gradients for mathematical expressions. This provides a the gradients at almost no cost of speed for (almost) any algebraic mathematical expression.

We achieved fully working implementation of a GP, with several covariance functions, which are the deciding factor of the form of the learned function (Linear, smooth, jagged, random walk etc.). We provided several notebooks showing the results of the Gaussian Process implementation.

- Linear: `http://nbviewer.ipython.org/github/mzwiessele/SymDiffPy/blob/master/notebooks/GP_example.ipynb`
- Non Linear: `http://nbviewer.ipython.org/github/mzwiessele/SymDiffPy/blob/master/notebooks/GP_linear_example.ipynb`
- Maona Loa CO2 levels: `http://nbviewer.ipython.org/github/mzwiessele/SymDiffPy/blob/master/notebooks/Mauna%20Loa%20C02.ipynb`

I thank my collaborators Meiwen, Victor, James and Daniel for their great work, I had a wonderful time implementing and experimenting with GPy and Theano.