proxpy: A python library for projection-based search

Projection-based methods of search have greater scope and are potentially more powerful than traditional methods of search. Unlike tree search, variables may be continuous, and unlike mathematical programming based methods, constraints are not limited to linear functions. The engines of projection-based methods are algorithms that find proximal points on constraint sets. These add flexibility to problem formulations while also leveraging the power of old algorithms in a novel setting. For example, the known efficient algorithms for singular value decomposition, or maximum-weight bipartite matching, are recruited when the search calls for the nearest rotation matrix, or the nearest permutation.

The proposed proxpy library tries to make projection-based methods accessible to a broad user base and thereby promote their development. Users would first formulate their problem as

$$\text{find } x \text{ such that:} \quad x \in A \cap B,$$

and then use constraint projections P_A and P_B from the library to iterate a map built from these that has the solution as an attractive fixed point. A particularly simple map, with a good track record on problems where at least one of the constraint sets is non-convex, is RRR:

$$x \mapsto x + \beta(P_A(2P_B(x) - x) - P_B(x)).$$

Here $\beta > 0$ is a parameter, analogous to the time-step of a dynamical system. A python interface for this approach to search makes sense because the elementary projections P_A and P_B provided by the library, that do all the heavy lifting, can be implemented efficiently in C (or another low level language) .

Spring 2018 is the official launch date of the development of proxpy. Below are the general guidelines.

Accessibility and participation

The package will be open source and hosted on github. Participation is initially limited to Cornell undergraduates, graduates, and postdocs. Work will be coordinated by Veit Elser in Physics. If you are interested, contact ve10@cornell.edu.

Name

Feel free to propose a different name. Alas, some good choices — aplpy, nearpy — have already been claimed.

Python programming

If you take pride in your python writing style and organization, there is certainly a role for you — even if you do not have expertise in C. We especially want the user interface/experience to be natural and familiar.

C programming

We definitely need a good C programmer for the core projections. For example, a much used projection is to the nearest positive semi-definite real (or hermitian) matrix of a given rank. Even GSL (GNU Scientific Library) does not offer an optimized version of this for the most used case, where the rank is much less than the size of the matrix. Code exists in the BLAS FORTRAN library, but setting up the linking will be a nightmare and limit portability. Ideally we need someone willing and able to write C programs such as this from scratch.

Software engineering

In the initial phase of the project we need someone with general software engineering expertise, a kind of chief-architect. We want the projections to be implemented efficiently but without compromising flexibility at the user interface. At the level of the RRR dynamical system the search variable x is just a vector of numbers (real or complex). However, each projection expects these same numbers to have a particular "shape" — a vector, matrix, tensor, edges of a bipartite graph — often to be compatible with data having the same shape. Instructions for shaping/linearizing should therefore also be part of the projection code-template and something that is easily specified by the user.

Documentation

The main document will be a guide to all the projections, with short usage examples. Veit Elser will write a tutorial on projection-based search featuring a broad cross-section of the projections.