`CREPE` manual

1. Introduction

`CREPE` stands for CRoss-Entropy Parameter Estimation. It is a tool used for global optimization, i.e. trying to find the underlying parameters of a model that best fit a set of constraints. In science, this kind of tool is usually applied when you have to deal with noisy data or when you have just a vague idea of what the parameters of a model should be, but don't know what they actually are. `CREPE` is based on the cross-entropy method attributed to Reuven Rubinstein. This method can be applied to a number of analyses, but this program focus on being a general tool for optimization.

2. Installation

`CREPE` is compatible with Python 2.7. Compatibility with other versions of Python is not yet verified. It requires the following packages:

- `numpy`
- `scipy`
- `nose`

The following packages are recommended:

- `matplotlib`: required to run the examples
- `pip`: used for easy uninstall

Assuming that you already downloaded `CREPE`, the installation can be done in the command line by using `$ python setup.py install`. To uninstall it, use `$ pip uninstall crepe`. You will probably need administrative rights to do so (use `sudo`).

3. Usage

3.1. `normal`

`normal` is the package that handles with independent parameters (i.e. each one is ruled by a single-variate normal distribution) and never touches the data. The most general method of including this package in your code is by using `from crepe import normal` in the beginning of your program.

3.1.1. optimize

`optimize` is a set of functions (hereafter *subpackage*) used to do the parameter estimation. The main function inside this subpackage is `estimate`. You, the user, has the job of feeding it with at least two inputs: an *initial guess* and a *performance function*. There are other optional inputs the user can feed (see the following subsections), but they're generally used when you have to deal with really noisy data or when you have no clue of what your parameters should be (i.e. the initial guess is too broad).

`crepe.normal.optimize.estimate(perf,p_mean,p_sigma,**kwargs)`:

`perf`: the performance function. It must be of the form $f(\boldsymbol{x})$, where $\boldsymbol{x}$ is a numpy array containing the current estimative of the parameters.
`p_mean`: numpy array containing the initial guess mean of each parameter.
`p_sigma`: numpy array containing the initial guess standard deviation of each parameter.

Optional parameters (`**kwargs`):
`N`: number of samples to be drawn. Default = 100
`rho`: sample lower quantile that will define the elite results. Default = 0.1
`c_limit`: lower limit for the relative change in the performance. Default = 0.1
`k_max`: maximum number of iterations. Default = 50
`alpha`: smoothing factor, between 0 and 1. Default = 1.0
`beta`: smoothing level, between 0.1 and 0.2 Default = 0.1
`silent`: `True` or `False`. Make it True to not print anything on the terminal. Default = `False`