Bayesian Optimization with scikit-optimize

Tim Head

31 August 2017

How to Make The Best

Beer recipes

Your task: brew the most tasty beer possible.

Many parameters you can tweak, for simplicity we let’s pretend there are only two: **alcohol content** and **bitterness**.

How do you find the best combination?

Evaluating a recipe is expensive

How to score a beer:

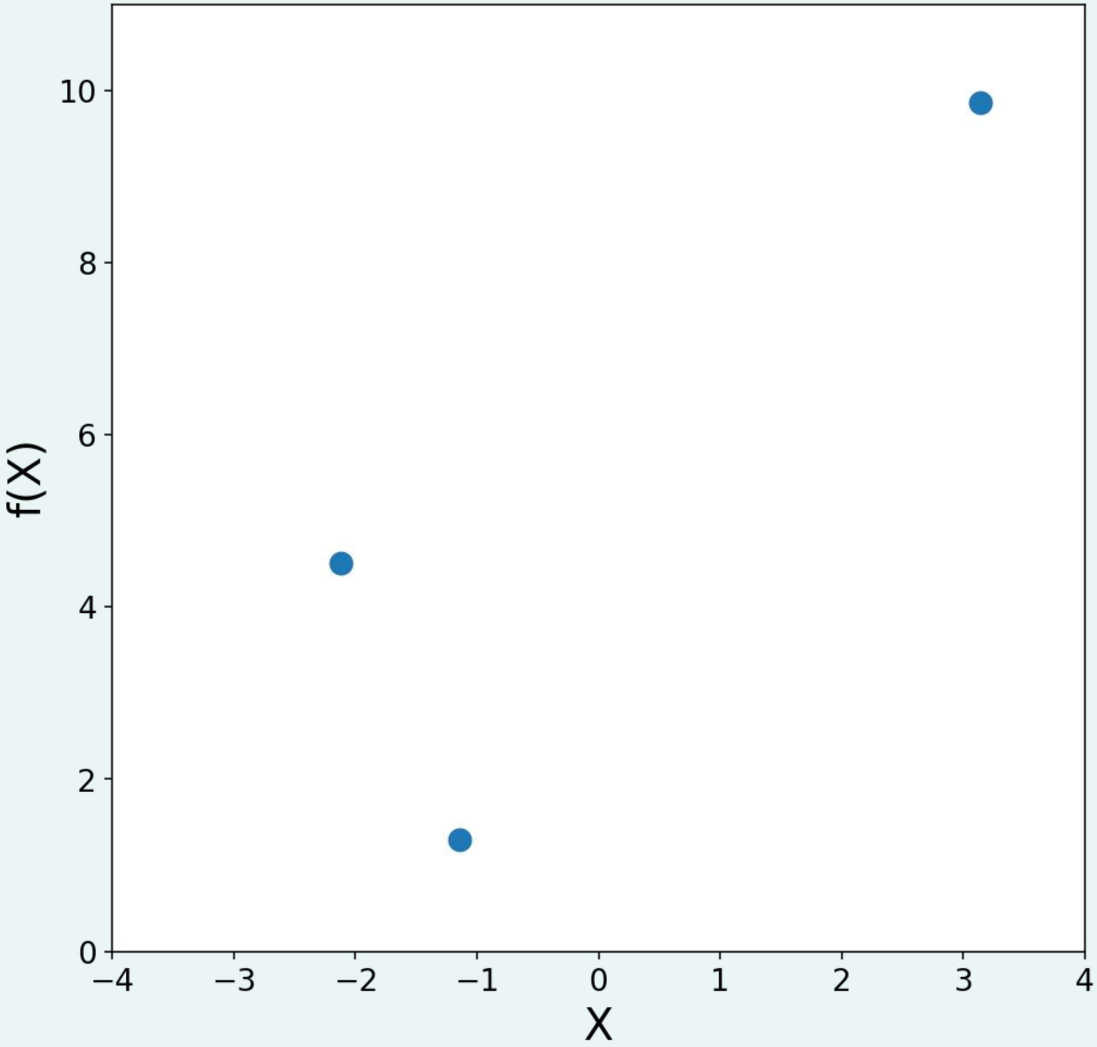
- Buy ingredients
- Brew it
- …wait…
- Find expert panel and collect scores

This means we can’t try a large number of combinations, have to be smart.

This is an optimization problem, with a (very) expensive *objective function*.

scipy.optimize!

Bayesian Optimisation


f(X)
X
0
2
4
6
8
10
−4
−3
−2
−1
0
1
2
3
4

Demo

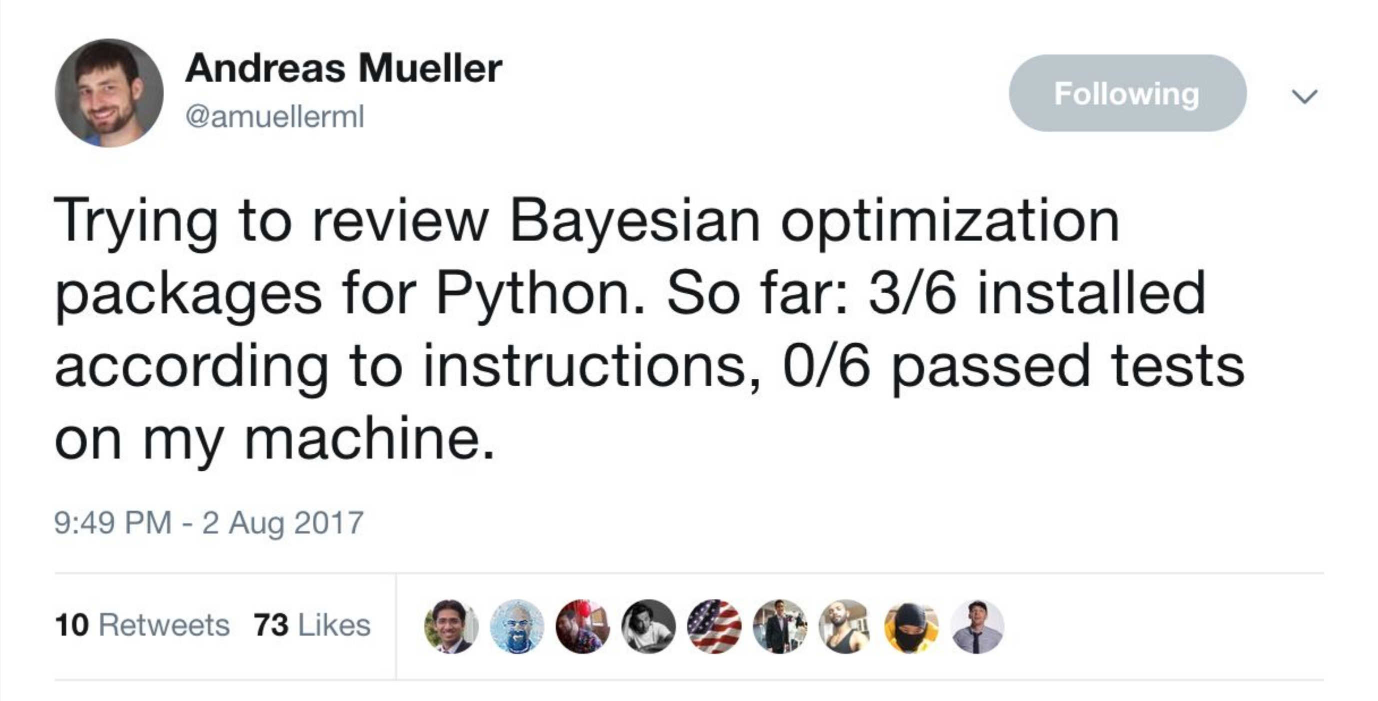
$$x^* = \arg\max_x f(x)$$

- f is a black box function, with no closed form nor gradients.
- f is expensive to evaluate.
- You only have noisy observations of f.

If you do not have these constraints, *do not* use Bayesian optimization.

Back to beer

scikit-optimize



Andreas Mueller
@amuellerml

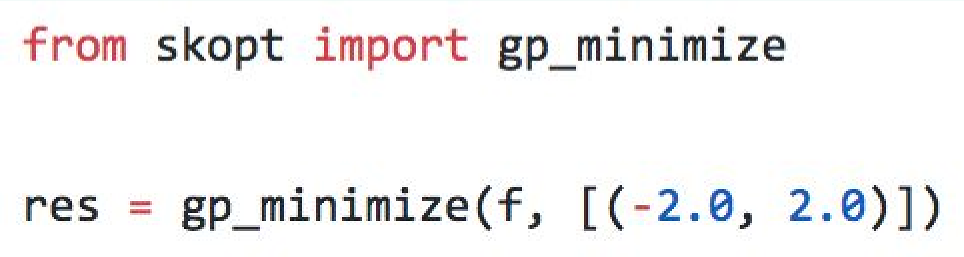
Trying to review Bayesian optimization packages for Python. So far: 3/6 installed according to instructions, 0/6 passed tests on my machine.

9:49 PM - 2 Aug 2017

10 Retweets **73** Likes

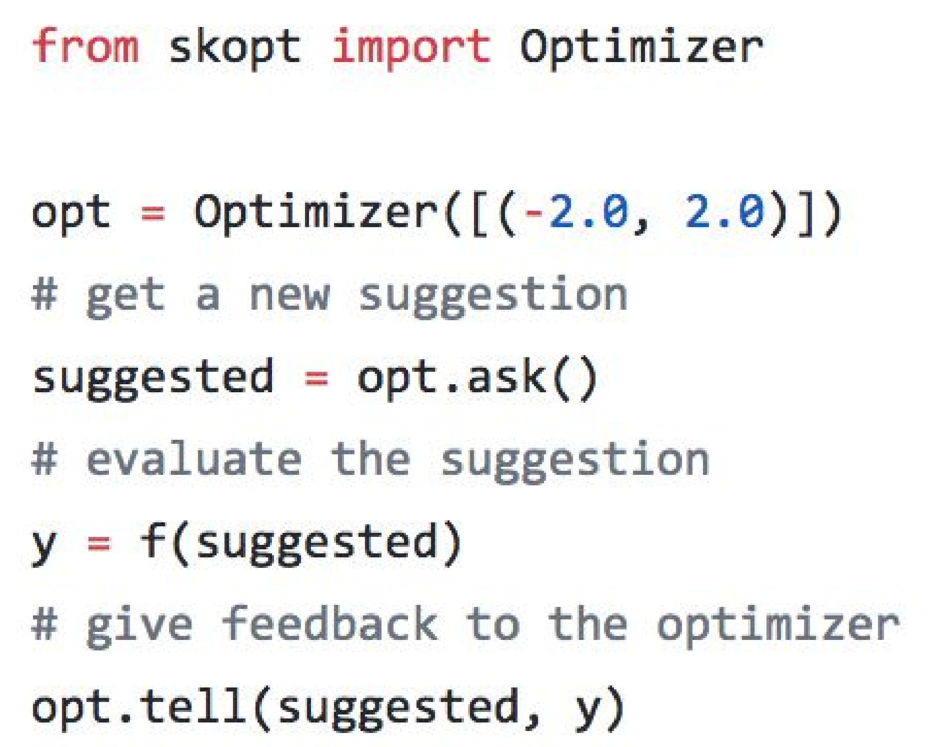
```
$ pip install numpy
$ pip install scikit-optimize
```

Like `scipy.optimize`



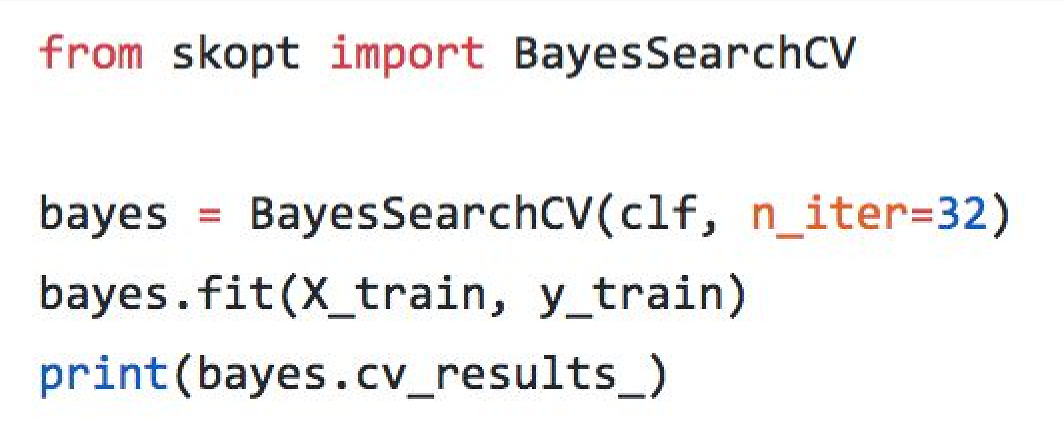
```
from skopt import gp_minimize


res = gp_minimize(f, [(-2.0, 2.0)])
```

Ask-and-tell interface

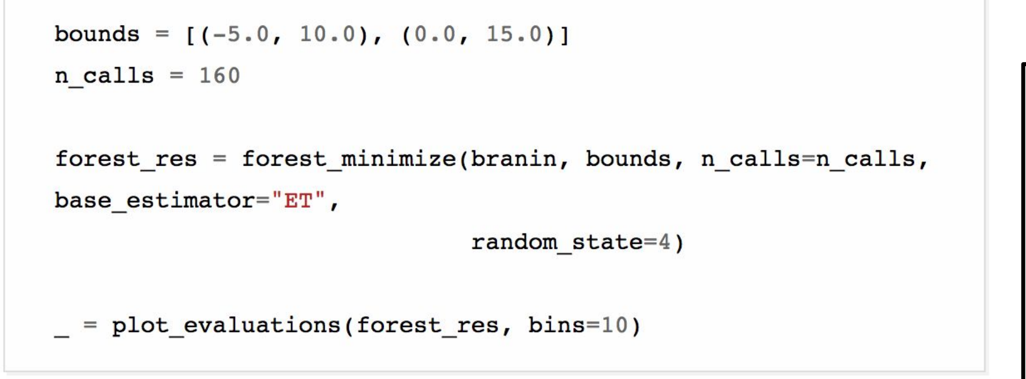


```
from skopt import Optimizer

opt = Optimizer([(-2.0, 2.0)])
# get a new suggestion
suggested = opt.ask()
# evaluate the suggestion
y = f(suggested)
# give feedback to the optimizer
opt.tell(suggested, y)
```

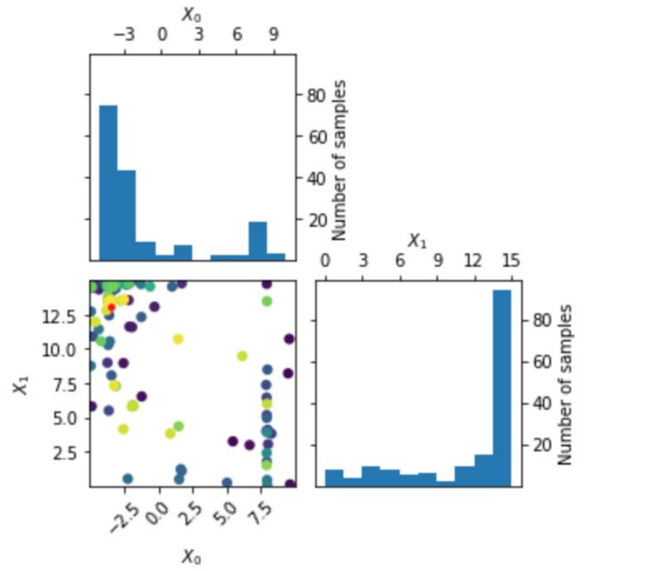
With scikit-learn

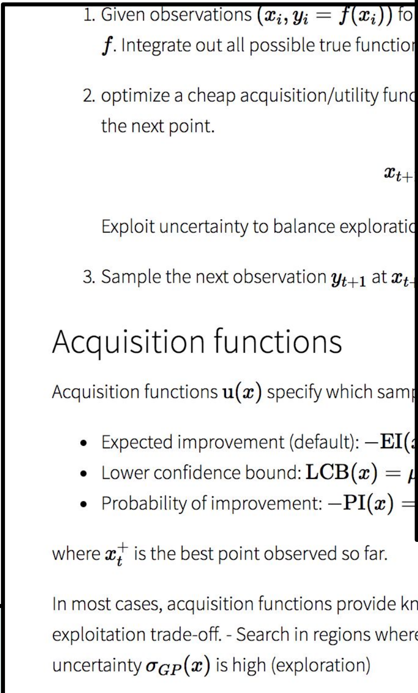


```
from skopt import BayesSearchCV


bayes = BayesSearchCV(clf, n_iter=32)
bayes.fit(X_train, y_train)
print(bayes.cv_results_)
```



```
bounds = [(-5.0, 10.0), (0.0, 15.0)]
n_calls = 160

forest_res = forest_minimize(branin, bounds, n_calls=n_calls,
base_estimator="ET",
                             random_state=4)

_ = plot_evaluations(forest_res, bins=10)
```





1. Given observations $(x_i, y_i = f(x_i))$ fo
f. Integrate out all possible true functio

2. optimize a cheap acquisition/utility func
the next point.

x_{t+}

Exploit uncertainty to balance exploratio

3. Sample the next observation y_{t+1} at x_{t+}

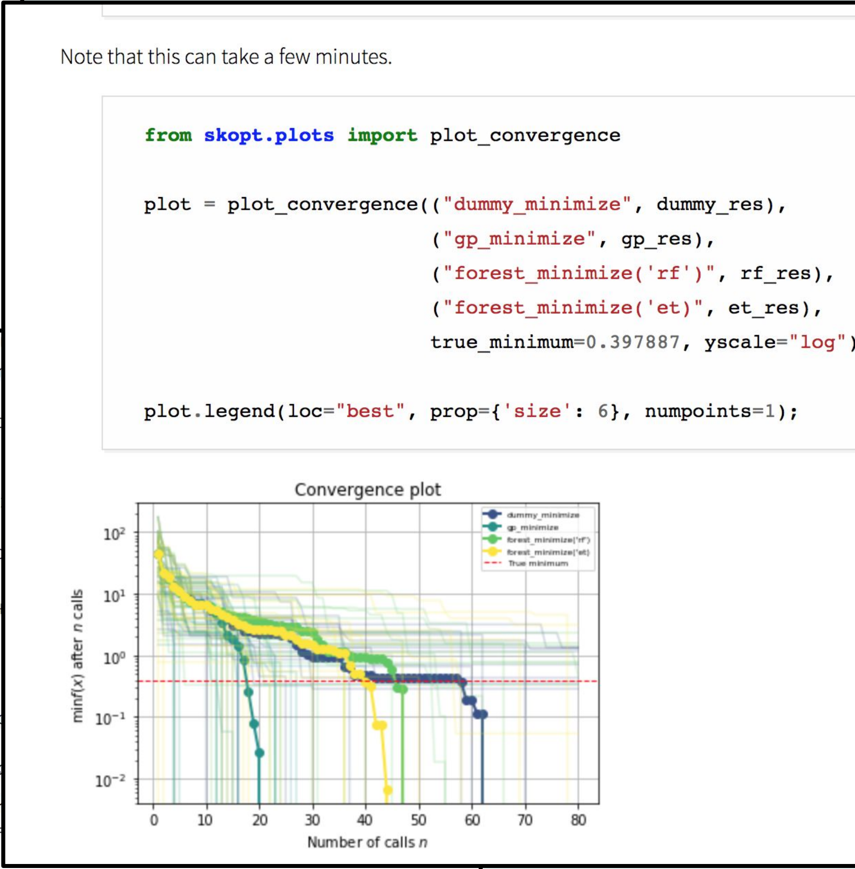
Acquisition functions

Acquisition functions $\mathbf{u}(x)$ specify which samp

- Expected improvement (default): $-\mathrm{EI}($
- Lower confidence bound: $\mathrm{LCB}(x) = \mu$
- Probability of improvement: $-\mathrm{PI}(x) =$

where x_t^+ is the best point observed so far.

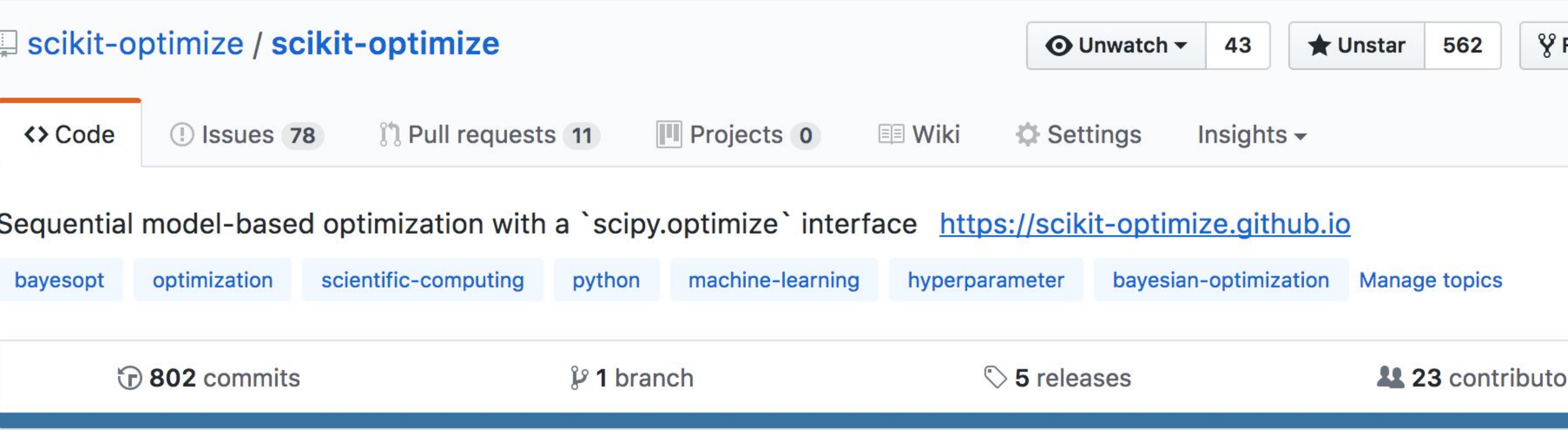
In most cases, acquisition functions provide knobs (e.g., κ) for controlling the exploration-exploitation trade-off. - Search in regions where $\mu_{GP}(x)$ is high (exploitation) - Probe regions where uncertainty $\sigma_{GP}(x)$ is high (exploration)



Note that this can take a few minutes.

```
from skopt.plots import plot_convergence

plot = plot_convergence(("dummy_minimize", dummy_res),
                        ("gp_minimize", gp_res),
                        ("forest_minimize('rf')", rf_res),
                        ("forest_minimize('et)", et_res),
                        true_minimum=0.397887, yscale="log")

plot.legend(loc="best", prop={'size': 6}, numpoints=1);
```

Join us (not just tomorrow)



Sequential model-based optimization with a `scipy.optimize` interface https://scikit-optimize.github.io

??!?

Tim Head
tim@wildtreetech.com
@betatim

To run the notebooks:
https://github.com/wildtreetech/bayesian-optimisation

Brewing beer is expensive and does not come with gradients, ***scikit-optimize can help.***