Citrine Applied Scientist Challenge Report

Xiaoting Zhong
xiaotinz@andrew.cmu.edu

1 Dependencies

- numpy: `https://scipy.org/install.html`
- networkx: `https://pypi.org/project/networkx/2.2/`

Note because I use anaconda, my python path is different from the default. In order to properly run my code, change the first line of the script from `!/Users/xiaotingzhong/anaconda2/bin/python` to `!/usr/local/bin/python`.

2 Algorithm Design

The classic algorithm for solving this kind of sampling problem is the Markov-chain Monte Carlo algorithm. If the feasible region is known, then one can discretize the space and perform random walk random walk on the discrete lattice [1, 2]. The difficulty of our problem comes from the non-linear constraints.

The key ideas in my algorithm design includes:

- Reorganize constraints into the order of involved variables so that constraints of a given variable can be easily checked. See `Constraint.get_var_constraints()` for details.
- Reduce the feasible region by checking linear constraints. The feasible region is recorded by a list called `bounds`, whose size is [n_dim, 2]. The first column of `bounds` records the low limits and the second column records the high limits. `bounds` is checked in two places. The first place is before sampling. The purpose is to take care of the single variable linear constraints. The second place is during sampling. The purpose is to check the multiple variable linear constraints if all variables in that constraint has been sampled. `bounds` is updated accordingly if linear constraint is passed.
- Group variables that share constraints and sample data from group to group. Note not all variables share constraints. In other words, variables that don't share constraints are independent of each other. We don't need to sample independent variables at the same time.
- For the schedule of sampling, a randomly picked variable is sampled from its feasible region time. In expectation, this strategy should be able to uniformly span the feasible region since the variables are independent and identically distributed. After a group of samples have all been sampled, check the constraints. If a constraint was violated, re-sample all variables appeared in that constraint. If all constraint satisfied, accept the data group and continue sample the next group until all data groups have been sampled.

Other possible improvements:

- Check of non-linear constraints. In current design, I only check non-linear constraints are the entire data tuple has been sampled. An alternative strategy is to check validity of a variables non-linear constraints after the sampling of this variable. This can be done

by using `scipy.optimize.minimize` if the function value is bounded. However, I ended up not using it because most function values in the examples are not bounded so `scipy.optimize.minimize` will fail.

- Schedule of re-sampling process. In current design, I started re-sampling several variables immediately after a constraint is violated. However, re-sample the last sampled data several times and check the violated constraint before start over is probably more efficient. This concept comes from the ideal of epsilon greedy [3].
- I also found a few papers on the topic of sampling data with non-linear constraints. However, wasn't able to digest them due to the time limit [4].

3 Performance

The performance of the code on the four given examples are reported below. Numbers listed in seconds.

	alloy	example	formulation	mixture
Layer Spec	3.8678	0.9308	176.0281	0.3884

References

[1] Luis Rademacher. Algorithmic convex geometry.

[2] Karel Van den Meersche, Karline Soetaert, and Dick Van Oevelen. xsample (): an r function for sampling linear inverse problems. *Journal of Statistical Software*, 30(Code Snippet 1), 2009.

[3] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[4] John W Chinneck. The constraint consensus method for finding approximately feasible points in nonlinear programs. *INFORMS Journal on Computing*, 16(3):255–265, 2004.