ComputeConf
User Documentation

December 10, 2010

1 Overview

`Computeconf` is a script in `coopr.pysp` to generate a confidence interval. It is based on Monte Carlo Sampling in stochastic programming and uses the multiple replication procedure (MRP).
To execute `computeconf` it is necessary to provide command line arguments. Which ones are needed in particular will be explained in Subsection 3.1 and Subsection 3.2. To get an overview of all command line arguments type
`computeconf --help`.
These command line arguments are also summarized in Section 4.

2 Monte Carlo Sampling in Stochastic Programming, MRP

Extract of [1]:

Monte Carlo Sampling

Suppressing the (fixed) decision x

Let $z = \mathbb{E} f(\xi)$, $\sigma^2 = \text{var} f(\xi) < \infty$ and $\xi^1, \ldots, \xi^n$ be i.i.d. as ξ

Let $z_n = \frac{1}{n} \sum_{i=1}^{n} f(\xi^i)$ be the sample mean estimator of z

FACT 1. $\boxed{E z_n = z}$ z_n is an *unbiased estimator* of z

FACT 2. $\boxed{z_n \to z, \text{ wp1}}$ (strong LLN) z_n is a *strongly consistent* estimator of z

FACT 3. $\boxed{\sqrt{n}(z_n - z) \Rightarrow N(0, \sigma^2)}$ (CLT) Rate of convergence is $1/\sqrt{n}$ and scaled difference is normally distributed

FACTS 4,5,... law of iterated logarithm, large deviations results,...

Monte Carlo Methods in Stochastic Programming

- True/population problem:

$$z^* = \min_{x \in X} \mathbb{E} f(x, \xi) \tag{SP}$$

  Denote optimal solution x^*

- Sample problem:

$$z_n^* = \min_{x \in X} \left[\frac{1}{n} \sum_{j=1}^{n} f(x, \xi^j) \right] \tag{SP$_n$}$$

  Here, $\xi^1, \ldots, \xi^n$ i.i.d. as ξ. Denote optimal solution x_n^*

- View z_n^* as an estimator of z^* and x_n^* as an estimator of x^*

- What *should* we want to say about z_n^* and x_n^* as $n \to \infty$?

Want names? *external sampling method*, *sample-path optimization*, *sample average approximation*, *stochastic counterpart*, *retrospective optimization*, and *non-recursive method*

Monte Carlo Methods in SP: *Possible* Goals

1. $x_n^* \to x^*$, wp1 and $\sqrt{n}(x_n^* - x^*) \Rightarrow N(0, \Sigma)$
2. $z_n^* \to z^*$, wp1 and $\sqrt{n}(z_n^* - z^*) \Rightarrow N(0, \sigma^2)$
3. $\mathbb{E} f(x_n^*, \xi) \to z^*$, wp1
4. $\lim_{n\to\infty} \mathbb{P}\left(\mathbb{E} f(x_n^*, \xi) - z^* \leq \varepsilon_n\right) \geq 1 - \alpha$ where $\varepsilon_n \to 0$

Modeling Issues:

• If (SP_n) is for maximum-likelihood estimation then goal 1 could be appropriate

• If (SP) is to price a financial option then goal 2 could be appropriate

• When (SP) is a decision-making model, 1 may be more then we need and 2 is of secondary interest. Goals 3 and 4 are arguably adequate

Technical Issues:

• In general, we shouldn't expect $\{x_n^*\}_{n=1}^\infty$ to converge when (SP) has multiple optimal solutions. In this case, we want: "limit points of $\{x_n^*\}_{n=1}^\infty$ solve (SP)"

• If we achieve "limit points" result, X is compact & $\mathbb{E} f(\cdot, \xi)$ is continuous then we obtain goal 3

• The limiting distributions may not be normal

Monte Carlo Methods in SP: Example

$$z^* = \min_{-1 \leq x \leq 1} \left[\mathbb{E} f(x, \xi) = \mathbb{E} \xi x\right], \text{ where } \xi \sim N(0,1)$$

Every feasible solution, $x \in [-1, 1]$ is optimal and $z^* = 0$

$$z_n^* = \min_{-1 \leq x \leq 1} \left(\frac{1}{n} \sum_{j=1}^n \xi^j\right) x$$

$x_n^* = \pm 1$, $z_n^* = -\left|N(0, 1/n)\right|$

Observations

1. $\mathbb{E} z_n^* \leq z^* \, \forall n$ (negative bias)
2. $\mathbb{E} z_n^* \leq \mathbb{E} z_{n+1}^* \, \forall n$ (monotonically shrinking bias)
3. $z_n^* \to z^*$, wp1 (strongly consistent)
4. $\sqrt{n}(z_n^* - z^*) = -\left|N(0,1)\right|$ (non-normal errors)
5. $b(z_n^*) \equiv \mathbb{E} z_n^* - z^* = a_1/\sqrt{n}$ ($O(n^{-1/2})$ bias)

So, *optimization* changes the nature of sample-mean estimators

Note: What if $x \in [-1, 1]$ is replaced with $x \in \mathbb{R}$?

Monte Carlo Methods in SP: Bias

$$\min_{x\in X} \mathbb{E}\left[\frac{1}{n}\sum_{j=1}^{n} f(x,\xi^j)\right] = \min_{x\in X} \mathbb{E}\, f(x,\xi) = z^*$$

and so we obtain

$$\mathbb{E} z_n^* = \mathbb{E}\left[\min_{x\in X}\frac{1}{n}\sum_{j=1}^{n} f(x,\xi^j)\right] \le \min_{x\in X} \mathbb{E}\, f(x,\xi) = z^*, \text{ i.e., } \boxed{\mathbb{E} z_n^* \le z^*}$$

Simple example when $n = 1$

$$\mathbb{E} \min_{x\in X} f(x,\xi) \le \min_{x\in X} \mathbb{E} f(x,\xi)$$

Interpretation: We'll do better if we get to "wait and see" ξ before choosing x

Also, can show bias decreases monotonically

$$\boxed{\mathbb{E} z_n^* \le \mathbb{E} z_{n+1}^* \le z^*}$$

Intuition...

Assessing Solution Quality

$$z^* = \min_{x\in X} \mathbb{E}\, f(x,\xi)$$

Our goal: Given $\hat{x} \in X$ and α find a (random) CI width ε with:

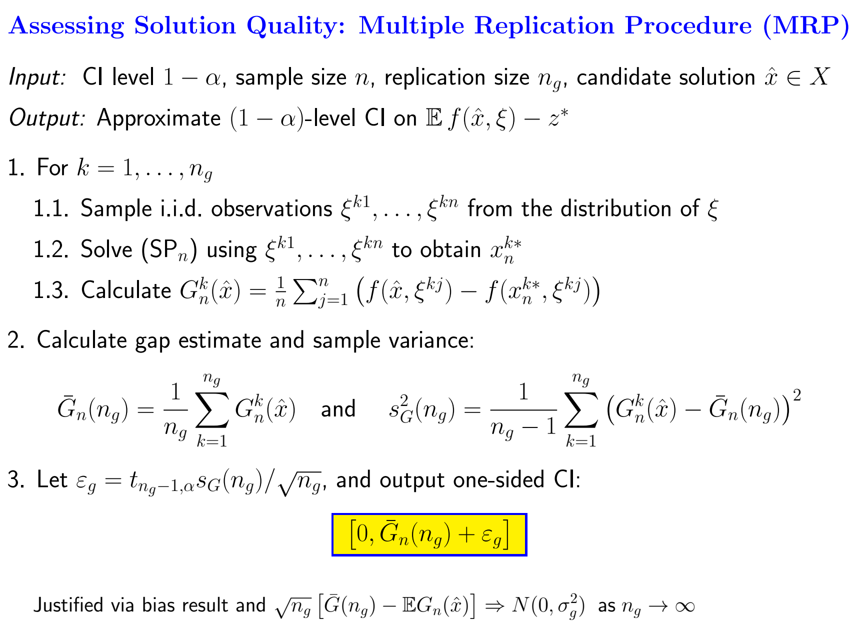
$$\boxed{\mathbb{P}(\mathbb{E}\, f(\hat{x},\xi) - z^* \le \varepsilon) \approx 1-\alpha}$$

Using the bias result,

$$\underbrace{\mathbb{E}\left[\frac{1}{n}\sum_{j=1}^{n} f(\hat{x},\xi^j) - \min_{x\in X}\frac{1}{n}\sum_{j=1}^{n} f(x,\xi^j)\right]}_{G_n(\hat{x})} \ge \mathbb{E} f(\hat{x},\xi) - z^*$$

Remarks

- Anticipate var $G_n(\hat{x}) \le$ var $[\frac{1}{n}\sum_{j=1}^{n} f(\hat{x},\xi^j)] +$ var z_n^*
- $G_n(\hat{x}) \ge 0$
- $G_n(\hat{x})$ is not asymptotically normal (what to do?)



Assessing Solution Quality: Multiple Replication Procedure (MRP)

Input: CI level $1-\alpha$, sample size n, replication size n_g, candidate solution $\hat{x} \in X$

Output: Approximate $(1-\alpha)$-level CI on $\mathbb{E}\, f(\hat{x}, \xi) - z^*$

```
1. For $k = 1, \dots, n_g$
   1.1. Sample i.i.d. observations $\xi^{k1}, \dots, \xi^{kn}$ from the distribution of $\xi$
   1.2. Solve (SP$_n$) using $\xi^{k1}, \dots, \xi^{kn}$ to obtain $x_n^{k*}$
   1.3. Calculate $G_n^k(\hat{x}) = \frac{1}{n}\sum_{j=1}^{n}\left(f(\hat{x}, \xi^{kj}) - f(x_n^{k*}, \xi^{kj})\right)$
2. Calculate gap estimate and sample variance:
```

$$\bar{G}_n(n_g) = \frac{1}{n_g}\sum_{k=1}^{n_g} G_n^k(\hat{x}) \quad \text{and} \quad s_G^2(n_g) = \frac{1}{n_g - 1}\sum_{k=1}^{n_g}\left(G_n^k(\hat{x}) - \bar{G}_n(n_g)\right)^2$$

```
3. Let $\varepsilon_g = t_{n_g-1,\alpha} s_G(n_g)/\sqrt{n_g}$, and output one-sided CI:
```

$$\boxed{\left[0, \bar{G}_n(n_g) + \varepsilon_g\right]}$$

Justified via bias result and $\sqrt{n_g}\left[\bar{G}(n_g) - \mathbb{E}G_n(\hat{x})\right] \Rightarrow N(0, \sigma_g^2)$ as $n_g \to \infty$

For more information about MRP, see [2].

3 Execute computeconf

`Computeconf` can be used for minimization problems as well as for maximization problems. Furthermore, there are two different ways to get the replicates for the MRP.

One way is to take a certain fraction of the provided scenarios to generate $\hat{x}$, i.e. $\hat{n}$ many, and devide the remaining scenarios in n_g groups of n scenarios. That means if N is the total number of scenarios then $N = \hat{n} + n \cdot n_g$.

Another way would be to provide n_g sets of each the same number of scenarios, i.e. n. That means we use one set of n scenarios to generate $\hat{x}$ and then we use n_g sets of each n scenarios to get the confidence intervall.

3.1 "Old" computeconf

To execute `computeconf` the old way, at least the following command line arguments are needed:

```
computeconf      --model-directory=MODEL_DIRECTORY
                 --instance-directory=INSTANCE_DIRECTORY
                 --number-samples-for-confidence-interval=N_G
                 --fraction-scenarios-for-solve=FRACTION_FOR_SOLVE
```

3.2 "New" computeconf

To execute `computeconf` the second way, least the following command line arguments are needed:

```
computeconf      --model-directory=MODEL_DIRECTORY
                 --instance-directory=INSTANCE_DIRECTORY
                 --number-samples-for-confidence-interval=N_G
                 --MRP-directory-basename=MRP_DIRECTORY_BASENAME
```

4 Command Line Arguments

Besides specific command line arguments for `computeconf`, one can also use arguments which are specified for `PH` to control the algorithm.

4.1 Computeconf Command Line Arguments

- `--fraction-scenarios-for-solve=FRACTION_FOR_SOLVE`
  The fraction of scenarios that are allocated to finding a solution. Default is 0.5.

- `--number-samples-for-confidence-interval=N_G`
  The number of samples of scenarios that are allocated to the confidence inteval (n_g). Default is 10.

- `--confidence-interval-alpha=CONFIDENCE_INTERVAL_ALPHA`
  The alpha level for the confidence interval. Default is 0.05.

- `--solve-xhat-with-ph`
  Perform xhat solve via PH rather than an EF solve. Default is False.

- `--random-seed=RANDOM_SEED`
  Seed the random number generator used to select samples. Defaults to 0, indicating time seed will be used.

- `--append-file=APPEND_FILE`
  File to which summary run information is appended, for output tracking purposes.

- `--write-xhat-solution`
  Write xhat solutions (first stage variables only) to the append file. Defaults to False.

- `--generate-weighted-cvar`
  Add a weighted CVaR term to the primary objective.

- `--cvar-weight=CVAR_WEIGHT`
  The weight associated with the CVaR term in the risk-weighted objective formulation. Default is 1.0. If the weight is 0, then *only* a non-weighted CVaR cost will appear in the EF objective - the expected cost component will be dropped.

- `--risk-alpha=RISK_ALPHA`
  The probability threshold associated with cvar (or any future) risk-oriented performance metrics. Default is 0.95.

- `--MRP-directory-basename=MRP_DIRECTORY_BASENAME`
  The basename for the replicate directories. It will be appended by the number of the group (loop over n_g). Default is None.

4.2 PH Command Line Arguments

- `--help, -h`
  Show help message and exit.

- `--verbose`
  Generate verbose output for both initialization and execution. Default is False.

- `--report-solutions`
  Always report PH weights prior to each iteration. Enabled if `--verbose` is enabled. Default is False.

- `--model-directory=MODEL_DIRECTORY`
  The directory in which all model (reference and scenario) definitions are stored. I.e., the ".py" files. Default is ".".

- `--instance-directory=INSTANCE_DIRECTORY`
  The directory in which all instances (reference and scenario) definitions are stored. I.e., the ".dat" files. Default is ".".

- `--solver=SOLVER_TYPE`
  The type of solver used to solve scenario sub-problems. Default is cplex.

- `--solver-manager=SOLVER_MANAGER_TYPE`
  The type of solver manager used to coordinate scenario sub-problem solves. Default is serial. This option is changed in parallel applications as described in [3].

- `--max-iterations=MAX_ITERATIONS`
  The maximal number of PH iterations. Default is 100.

- `--default-rho=DEFAULT_RHO`
  The default (global) rho for all blended variables. Default is 1.

- `--rho-cfgfile=RHO_CFGFILE`
  The name of a configuration script to compute PH rho values. Default is None.

- `--enable-termdiff-convergence`
  Terminate PH based an the termdiff convergence metric. The convergence metric is the unscaled sum of differences between variable values and the mean. Default is True.

- `−−enable-normalized-termdiff-convergence`
  Terminate `PH` based on the normalized termdiff convergence metric. Each term in the termdiff sum is normalized by the average value (NOTE: it is NOT normalized by the number of scenarios). Default is False.

- `−−termdiff-threshold=TERNDIFF_THRESHOLD`
  The convergence threshold used in the term-diff and normalized term-diff convergence criteria. Default is 0.01, which is too low for most problems.

- `−−enable-free-discrete-count-convergence`
  Terminate PH based on the free discrete variable count convergence metric. Default is False.

- `−−free-discrete-count-threshold=FREE_DISCRETE_COUNT_THRESHOLD`
  The convergence threshold used in the criterion based on when the free discrete variable count convergence criterion. Default is 20.

- `−−enable-ww-extensions`
  Enable the Watson-Woodruff `PH` extensions plugin. Default is False.

- `−−ww-extension-cfgfile=WW_EXTENSION_CFGFILE`
  The name of a configuration file for the Watson-Woodruff `PH` extensions plugin. Default is wwph.cfg.

- `−−ww-extension-suffixfile=WW_EXTENSION_SUFFIXFILE`
  The name of a variable suffix file for the Watson-Woodruff `PH` extensions plugin. Default is wwph.suffixes.

- `−−user-defined-extension=EXTENSIONFILE`
  Here, "EXTENSIONFILE" is the module name, which is in either the current directory (most likely) or somewhere on your PYTHONPATH. A simple example is "testphextension" plugin that simply prints a message to the screen for each callback. The file testphextension.py can be found in the sources directory. A test of this would be to specify "`−−user-defined-extension=testphextension`", assuming testphextension.py is in your PYTHONPATH or current directory. Note that both `PH` extensions (WW PH and your own) can co-exist; however, the WW plugin will be invoked first.

- `−−scenario-solver-options`
  The options are specified just as in pyomo, e.g.,
  `−−scenario-solver-options="mip_tolerances_mipgap=0.2"` to set the mipgap for all scenario sub-problem solves to 20% for the CPLEX solver. The options are specified in a quote delimited string that is passed to the sub-problem solver. Whatever options specified are persistent across all solves.

- `−−ef-solver-options`
  The options are specified just as in pyomo, e.g.,
  `−−ef-solver-options="mip_tolerances_mipgap=0.2"` to set the mipgap for all scenario sub-problem solves to 20% for the CPLEX solver. The options are specified in a quote delimited string that is passed to the `EF` problem solver.

- `--write-ef`
  Upon termination, write the extensive form of the model - accounting for all fixed variables.

- `--solve-ef`
  Following write of the extensive form model, solve it.

- `--ef-output-file=EF_OUTPUT_FILE` The name of the extensive form output file (currently only LP format is supported), if writing of the extensive form is enabled. Default is efout.lp.

- `--suppress-continuous-variable-output`
  Eliminate `PH`-related output involving continuous variables. Default: no output.

- `--keep-solver-files`
  Retain temporary input and output files for scenario sub-problem solves. Default: files not kept.

- `--output-solver-logs` Output solver logs during scenario sub-problem solves. Default: no output.

- `--output-ef-solver-log`
  Output solver log during the extensive form solve. Default: no output.

- `--output-solver-results`
  Output solutions obtained after each scenario sub-problem solve. Default: no output.

- `--output-times`
  Output timing statistics for various `PH` components. Default: no output.

- `--disable-warmstarts`
  Disable warm-start of scenario sub-problem solves in `PH` iterations. Default is False (i.e., warm starts are the default).

- `--drop-proximal-terms`
  Eliminate proximal terms (i.e., the quadratic penalty terms) from the weighted `PH` objective. Default is False (i.e., but default, the proximal terms are included).

- `--retain-quadratic-binary-terms`
  Do not linearize `PH` objective terms involving binary decision variables. Default is False (i.e., the proximal term for binary variables is linearized by default; this can have some impact on the relaxations during the branch and bound solution process).

- `--linearize-nonbinary-penalty-terms=BPTS`
  Approximate the `PH` quadratic term for non-binary variables with a piecewise linear function. The argument BPTS gives the number of breakpoints in the linear approximation. The argument BPTS gives the number of breakpoints in the linear approximation. The default is 0. Reasonable non-zero values are usually in the range of 3 to 7. Note that if a breakpoint would be very close to a variable bound, then the break point is ommited. IMPORTANT: this option requires that all variables have bounds

that are established in the reference model or by code specified using the bounds-cfgfile command line option.

- `--breakpoint-strategy=BREAKPOINT_STRATEGY`
  Specify the strategy to distribute breakpoints on the [lb, ub] interval of each variable when linearizing. 0 indicates uniform distribution. 1 indicates breakpoints at the node min and max, uniformly in between. 2 indicates more aggressive concentration of breakpoints near the observed node min/ max.

- `--bounds-cfgfile=BOUND_CFGFILE`
  The argument BOUNDS_CFGFILE specifies the name of an executable pyomo file that sets bounds. The default is that there is no file. When specified, the code in this file is executed after the initialization of scenario data so the bounds can be based on data from all scenarios.

- `--checkpoint-interval`
  The number of iterations between writing of a checkpoint file. Default is 0, indicating never.

- `--restore-from-checkpoint`
  The name of the checkpoint file from which `PH` should be initialized. Default is not to restore from a checkpoint.

- `--profile=PROFILE`
  Enable profiling of Python code. The value of this option is the number of functions that are summarized. The default is no profiling.

- `--enable-gc`
  Enable the python garbage collecter. The default is no garbage collection.

5 Examples

5.1 Example for the "old" way

5.2 Example for the "new" way

References

[1] G. Bayraksan and D. Morton. Monte Carlo Sampling in Stochastic Programming: Assessing Solution Quality and Sequential Sampling.

[2] W.-K. Mak, D. P. Morton and R. K. Wood. Monte Carlo bounding techniques for determining solution quality in stochastic programs. October 1998

[3] J.-P. Watson and D. L. Woodruff. PYSP Version 1.1, User Documentation. January 2010