

Risk Measures in Bayesian Optimization

Overview

Risk measures transform a distribution of outcomes into a single scalar that captures **how bad things can get** under uncertainty. In BO, the uncertainty comes from environmental/context variables (manufacturing tolerances, operating conditions, etc.) that the optimizer cannot control.

1. Single-Objective Risk Measures

Value-at-Risk (VaR)

The largest value that the objective exceeds with probability $\geq \alpha$.

$$\text{VaR}_\alpha[f(\mathbf{x} \diamond \boldsymbol{\xi})] = \sup\{z \in \mathbb{R} : P[f(\mathbf{x} \diamond \boldsymbol{\xi}) \geq z] \geq \alpha\}$$

In discrete MC approximation with n_w samples sorted descending:

$$\text{VaR}_\alpha \approx Y_{(\lceil \alpha n_w \rceil)} \quad (\text{the } \lceil \alpha n_w \rceil\text{-th largest value})$$

- $\alpha = 0.8$: the value exceeded by 80% of scenarios — higher α is *more* conservative.
- A single point estimate — tells you *where* the bad tail starts, not *how bad* it gets.

Convention note: The paper (Daulton et al., 2022) defines α as a confidence level where higher α = more conservative. BoTorch's internal `alpha` parameter is inverted: higher `alpha` = less conservative (fraction of samples to keep). `foamBO` maps between them via `alpha_botorch = max(1 - robustness, 0.05)`.

Conditional Value-at-Risk (CVaR)

The **average** outcome in the worst $(1 - \alpha)$ fraction of scenarios.

$$\text{CVaR}_\alpha(Y) = \frac{1}{n_w - k} \sum_{i \in \text{worst}} Y_i, \quad k = \lceil n_w \cdot \alpha \rceil$$

```

flowchart LR
  A["n_w posterior\nsamples"] --> B["Sort ascending"]
  B --> C["Take bottom\nn_w - k values"]
  C --> D["Mean -> CVaR"]
  style D fill:#f96,stroke:#333

```

- $\alpha \rightarrow 1 \rightarrow$ nearly risk-neutral (\approx Expectation)
- $\alpha \rightarrow 0 \rightarrow$ extremely conservative (\approx WorstCase)
- $\text{CVaR}_\alpha \leq \text{VaR}_\alpha$ always — it captures the *severity* of the tail, not just its boundary.

Comparison Table

| Measure | Formula | Conservatism | Differentiable? |
|-------------|---|---------------------|-----------------|
| Expectation | $\frac{1}{n_w} \sum_i Y_i$ | None (risk-neutral) | ✓ |
| VaR | $\sup\{z : P[Y \geq z] \geq \alpha\}$ | Moderate | ✗ (step fn) |
| CVaR | $\frac{1}{(1-\alpha)n_w} \sum_{i \in \text{worst}} Y_i$ | High | ✓ (smooth) |
| WorstCase | $\min(Y_1, \dots, Y_{n_w})$ | Maximum | ✗ |

Why CVaR is preferred: differentiable + captures tail severity. VaR only marks the boundary; CVaR tells you what happens beyond it.

2. Multi-Objective Risk Measures

When optimizing multiple objectives simultaneously, risk measures become set-valued.

IndependentCVaR

Applies CVaR **separately** to each objective.

$$\text{IndependentCVaR}_\alpha(\mathbf{Y}) = [\text{CVaR}_\alpha(Y_1), \text{CVaR}_\alpha(Y_2), \dots, \text{CVaR}_\alpha(Y_m)]$$

- Simple but **overestimates robustness** — ignores correlations between objectives.
- The worst scenarios for objective 1 may differ from worst scenarios for objective 2.

MVaR (Multivariate Value-at-Risk)

The **set** of non-dominated lower bounds that the objectives exceed jointly with probability $\geq \alpha$ (Prekopa, 2012).

$$\text{MVaR}_\alpha[\mathbf{f}(\mathbf{x} \diamond \boldsymbol{\xi})] = \text{Pareto}(\{\mathbf{z} \in \mathbb{R}^M : P[\mathbf{f}(\mathbf{x} \diamond \boldsymbol{\xi}) \geq \mathbf{z}] \geq \alpha\})$$

The **global MVaR** across the design space (analogous to the Pareto frontier for robust MO):

$$\text{MVaR}_\alpha[\{\mathbf{f}(\mathbf{x} \diamond \boldsymbol{\xi})\}_{\mathbf{x} \in X}] = \text{Pareto}(\bigcup_{\mathbf{x} \in X} \text{MVaR}_\alpha[\mathbf{f}(\mathbf{x} \diamond \boldsymbol{\xi})])$$

- Set-valued → not directly usable as an acquisition objective.
- Expensive to compute: time complexity exponential in M , size exponential in $|\text{MVaR}_\alpha|$.

MARS (MVaR Approximation via Random Scalarizations)

Daulton et al., "Robust Multi-Objective BO under Input Noise," ICML 2022.

MARS exploits a theoretical bijection between MVaR points and Chebyshev scalarization weights to approximate the full MVaR set via single-objective BO with random scalarizations.

Theorem 5.1 (MVaR \iff VaR of Chebyshev scalarization)

Under mild assumptions (continuous, strictly increasing CDF — satisfied by GP priors), there is a **bijection** between MVaR points and scalarization weights:

$$h : \text{MVaR}_\alpha[\mathbf{f}(\mathbf{x} \diamond \boldsymbol{\xi})] \rightarrow \Delta_+^{M-1}, \quad h(\mathbf{z}) = \mathbf{w} = \frac{\mathbf{z}}{\|\frac{1}{2}\mathbf{z}\|_1}$$

$$h^{-1}(\mathbf{w}) = \mathbf{z} = \frac{\mathbf{v}}{\mathbf{w}}, \quad \mathbf{v} = \text{VaR}_\alpha(s[\mathbf{f}(\mathbf{x} \diamond \boldsymbol{\xi}), \mathbf{w}])$$

This means **every point in the MVaR set corresponds to a unique weight vector**, and recovering it requires only computing VaR of a scalar Chebyshev scalarization.

Chebyshev scalarization

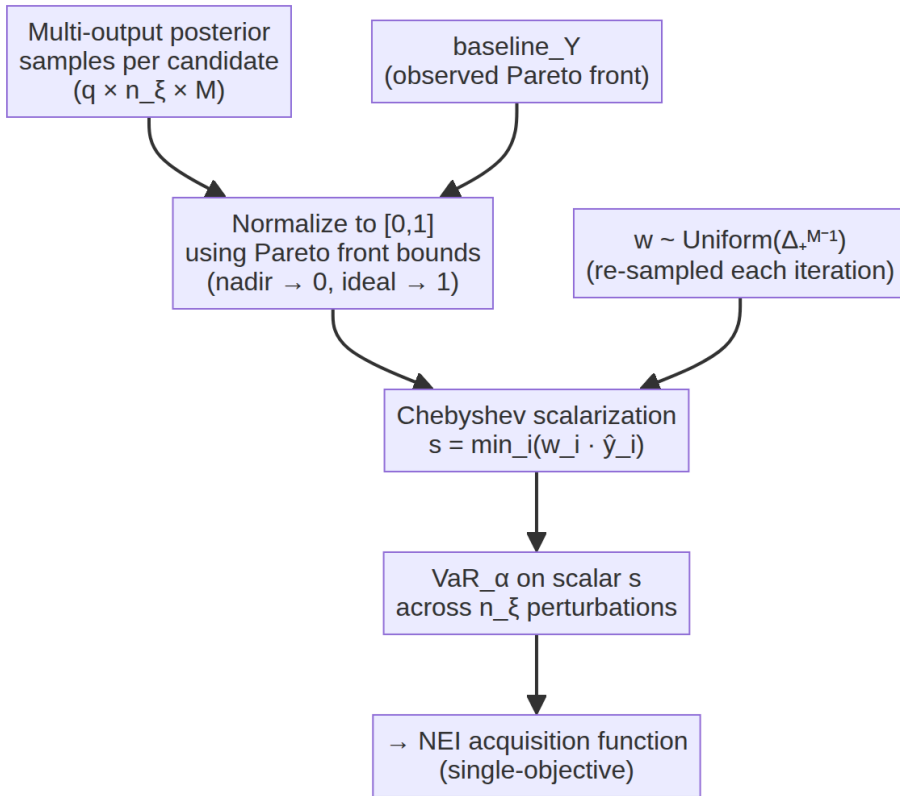
$$s[\mathbf{y}, \mathbf{w}, \mathbf{r}] = \min_i w_i (y_i - r_i)$$

where $\mathbf{w} \in \Delta_+^{M-1}$ (positive simplex) and $\mathbf{r} \in \mathbb{R}^M$ is the reference point. After normalizing \mathbf{f} to the unit cube, $\mathbf{r} = \mathbf{0}$ and we get $s[\mathbf{y}, \mathbf{w}] = \min_i (w_i y_i)$.

MARS algorithm

At each BO iteration:

1. Sample \mathbf{w} **uniformly** from the simplex Δ_+^{M-1}
2. Optimize $\text{VaR}_\alpha(s[\mathbf{f}(\mathbf{x} \diamond \boldsymbol{\xi}), \mathbf{w}])$ using a single-objective acquisition function (NEI)
3. Over iterations, different \mathbf{w} vectors trace out the full MVaR frontier



Key equations:

$$\hat{Y}_m = \frac{Y_m - \text{nadir}_m}{\text{ideal}_m - \text{nadir}_m} \quad (\text{normalize to } [0, 1])$$

$$s[\mathbf{Y}, \mathbf{w}] = \min_m (w_m \cdot \hat{Y}_m) \quad (\text{Chebyshev scalarization})$$

$$\text{MARS}(\mathbf{x}, \mathbf{w}) = \text{VaR}_\alpha(s[\mathbf{f}(\mathbf{x} \diamond \xi_1), \mathbf{w}], \dots, s[\mathbf{f}(\mathbf{x} \diamond \xi_{n_\xi}), \mathbf{w}])$$

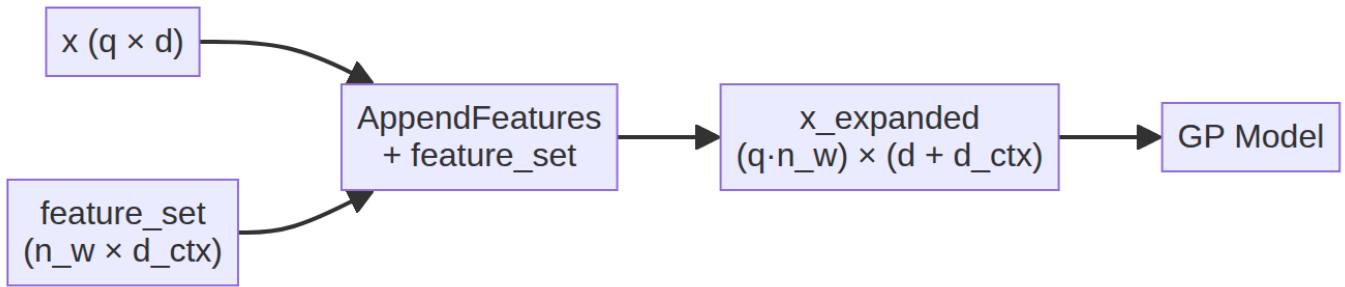
Why Chebyshev? The \min operation ensures all objectives must be good simultaneously — a design that excels in one objective but fails another scores poorly. The bijection (Theorem 5.1) guarantees that optimizing VaR of this scalarization recovers a globally optimal MVAR point (Corollary 5.1 in the paper).

3. The Input Expansion Mechanism

Risk measures need n_w scenarios per candidate. Two BoTorch transforms create them:

AppendFeatures (for context variables)

Used when environmental variables are **separate dimensions** from design variables.



Input: $x = [x_1, x_2]$ (design variables)
 Output: $[x_1, x_2, c_1^{(1)}, c_2^{(1)}]$ context scenario 1
 $[x_1, x_2, c_1^{(2)}, c_2^{(2)}]$ context scenario 2
 ...
 $[x_1, x_2, c_1^{(n)}, c_2^{(n)}]$ context scenario n_w

InputPerturbation (for noisy design variables)

Used when the uncertainty is additive/multiplicative noise on the design variables themselves.

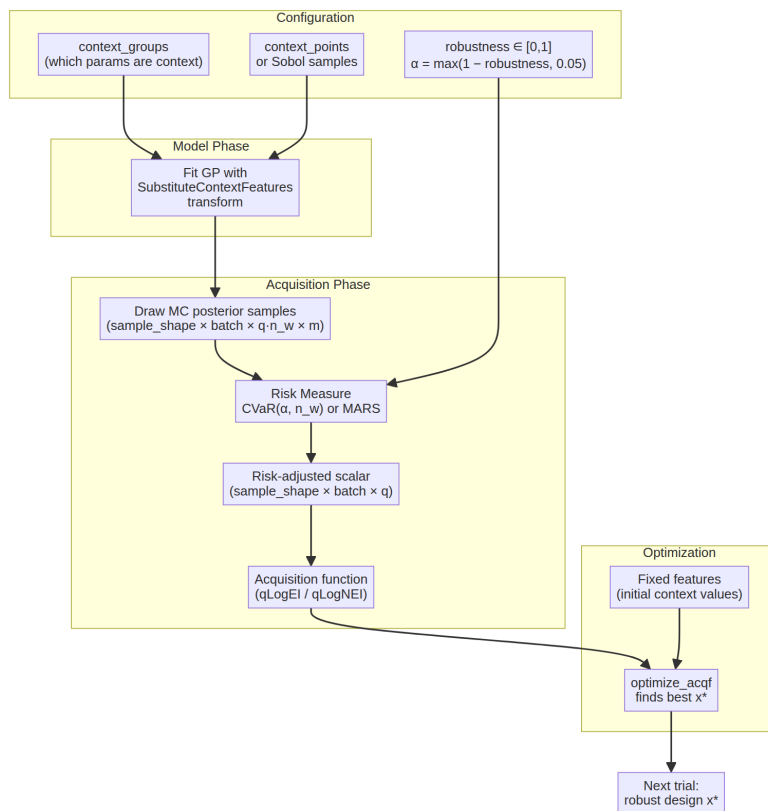
Input: $x = [x_1, x_2]$
 Output: $[x_1 + \delta_1^{(1)}, x_2 + \delta_2^{(1)}]$ perturbation 1
 $[x_1 + \delta_1^{(2)}, x_2 + \delta_2^{(2)}]$ perturbation 2
 ...

foamBO uses: SubstituteContextFeatures

A custom variant that **replaces** (not appends) context dimensions with scenario values.

Input: $x = [x_1, x_2, c_1, c_2]$ (design + context in same space)
 Output: $[x_1, x_2, c_1^{(1)}, c_2^{(1)}]$ context dims substituted
 $[x_1, x_2, c_1^{(2)}, c_2^{(2)}]$
 ...

4. Complete Pipeline



Step-by-step:

- 1. Expand** — Input transform replicates each candidate x across n_w context scenarios
- 2. Predict** — GP gives posterior samples at all $q \times n_w$ points
- 3. Reshape** — Risk measure groups samples: $(q \times n_w) \rightarrow q \times n_w$
- 4. Aggregate** — CVaR/VaR/MARS collapses the n_w dimension to one scalar per q
- 5. Optimize** — Acquisition function maximizes the risk-adjusted objective

5. SAASBO vs Risk Measures

SAASBO (Sparse Axis-Aligned Subspace BO) and risk measures solve **different problems**.

| Aspect | SAASBO | Risk Measures (CVaR/MARS) |
|----------------|-----------------------------------|--|
| Problem | High-dimensional design space | Uncertain operating conditions |
| Mechanism | Sparsity prior on GP lengthscales | Tail-averaging over context scenarios |
| Effect | Finds which parameters matter | Finds designs robust to what can't be controlled |
| Model | Fully Bayesian GP (NUTS sampling) | Standard GP + input transform |
| Complementary? | ✓ Can combine both | ✓ Can combine both |

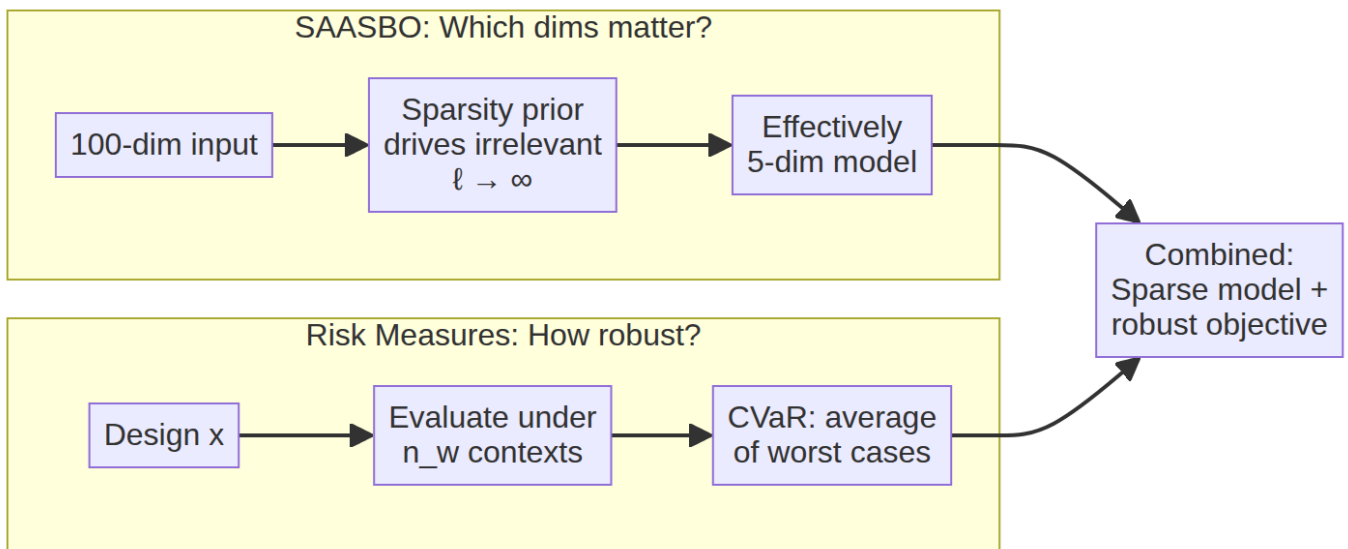
SAASBO Sparsity Prior

$$\tau^2 \sim \text{HalfCauchy}(0.1) \quad (\text{global shrinkage})$$

$$\lambda_i^2 \sim \text{HalfCauchy}(1) \quad (\text{per-dimension})$$

$$\ell_i = \frac{1}{\sqrt{\tau^2 \cdot \lambda_i^2}}$$

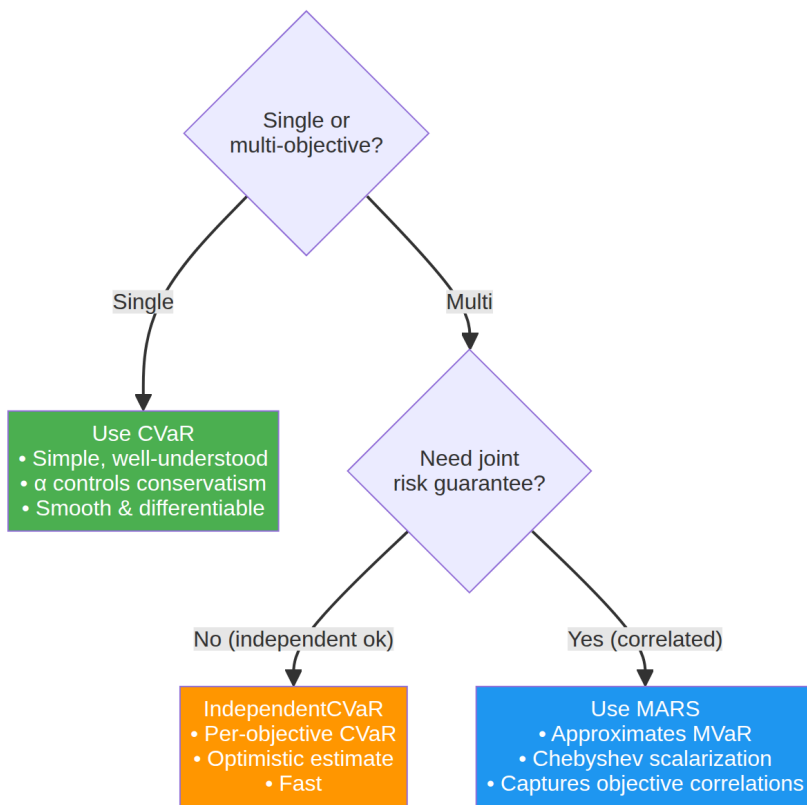
- Small $\tau^2 \rightarrow$ most $\ell_i \rightarrow \infty$ (irrelevant dimensions)
- Large λ_i^2 can overcome shrinkage (important dims keep finite ℓ_i)



CONTEXT_SACBO (Ax)

CONTEXT_SACBO was an Ax Models enum that combined SAASBO with context variables. It is **deprecated** — Ax now maps it to BOTORCH_MODULAR. foamBO does not use it; instead, it wires context handling directly via SubstituteContextFeatures + CVaR.

6. CVaR vs MARS: When to Use Which



| Criterion | CVaR | MARS |
|---------------------|------------------------|---|
| Objectives | Single | Multi |
| Output | Scalar | Scalar (via scalarization) |
| Risk correlation | N/A | Captures via joint scalarization |
| Requires baseline_Y | No | Yes (Pareto front for normalization) |
| Complexity | $O(n_w \log n_w)$ sort | $O(n_w \log n_w)$ + Chebyshev + normalize |
| foamBO support | ✓ Fully wired | Available in BoTorch, not yet wired |

Robustness Parameter Mapping (foamBO)

$$\alpha = \max(1 - \text{robustness}, 0.05), \quad \text{robustness} \in [0, 1]$$

| robustness | α | Behavior |
|------------|----------|------------------------------------|
| 0.0 | 1.00 | CVaR \approx E[Y] (risk-neutral) |
| 0.5 | 0.50 | Mean of worst 50% |
| 0.8 | 0.20 | Mean of worst 80% |
| 0.95 | 0.05 | Mean of worst 95% (capped) |

7. Summary Equations

$$\text{Expectation: } E[Y] = \frac{1}{n_w} \sum_{i=1}^{n_w} Y_i$$

$$\text{VaR}_\alpha : \sup\{z : P[Y \geq z] \geq \alpha\} \approx Y_{(\lceil \alpha n_w \rceil)} \text{ (sorted descending)}$$

$$\text{CVaR}_\alpha : \frac{1}{(1-\alpha)n_w} \sum_{i \in W} Y_i, \quad W = \text{worst } (1-\alpha)n_w \text{ outcomes}$$

$$\text{WorstCase: } \min(Y_1, \dots, Y_{n_w})$$

$$\text{MARS: } \text{VaR}_\alpha(\min_m(w_m \cdot \hat{Y}_m)), \quad \hat{Y} = \frac{Y - \text{nadir}}{\text{ideal} - \text{nadir}}, \quad \mathbf{w} \sim \text{Uniform}(\Delta_+^{M-1})$$

$$\text{SAAS prior: } \ell_i = \frac{1}{\sqrt{\tau^2 \cdot \lambda_i^2}}, \quad \tau^2 \sim \text{HC}(0.1), \quad \lambda_i^2 \sim \text{HC}(1)$$