16 Surrogate Optimization

The previous chapter explained how to use a probabilistic surrogate model, in particular a Gaussian process, to infer probability distributions over the true objective function. These distributions can be used to guide an optimization process toward better design points.[1] This chapter outlines several common techniques for choosing which design point to evaluate next. The techniques we discuss here greedily optimize various metrics.[2] We will also discuss how surrogate models can be used to optimize an objective measure in a safe manner.

[1] A. Forrester, A. Sobester, and A. Keane, *Engineering Design via Surrogate Modelling: A Practical Guide*. Wiley, 2008.

[2] An alternative to greedy optimization is to frame the problem as a *partially observable Markov decision process* and plan ahead some number of steps as outlined by M. Toussaint, "The Bayesian Search Game," in *Theory and Principled Methods for the Design of Metaheuristics*, Y. Borenstein and A. Moraglio, eds. Springer, 2014, pp. 129–144. See also R. Lam, K. Willcox, and D. H. Wolpert, "Bayesian Optimization with a Finite Budget: An Approximate Dynamic Programming Approach," in *Advances in Neural Information Processing Systems* (*NIPS*), 2016.

16.1 Prediction-Based Exploration

In *prediction-based exploration*, we select the minimizer of the surrogate function. An example of this approach is the quadratic fit search that we discussed earlier in section 3.5. With quadratic fit search, we use a quadratic surrogate model to fit the last three bracketing points and then select the point at the minimum of the quadratic function.

If we use a Gaussian process surrogate model, prediction-based optimization has us select the minimizer of the mean function

$$\mathbf{x}^{(m+1)} = \underset{\mathbf{x}\in\mathcal{X}}{\arg\min}\ \hat{\mu}(\mathbf{x}) \tag{16.1}$$

where $\hat{\mu}(\mathbf{x})$ is the predicted mean of a Gaussian process at a design point $\mathbf{x}$ based on the previous m design points. The process is illustrated in figure 16.1.

Prediction-based optimization does not take uncertainty into account, and new samples can be generated very close to existing samples. Sampling at locations where we are already confident in the objective value is a waste of function evaluations.

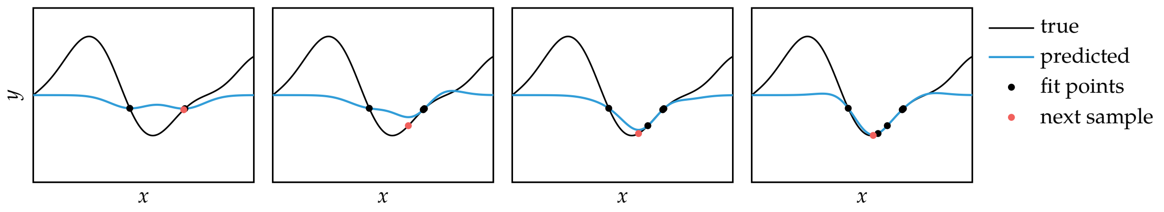


Figure 16.1. Prediction-based optimization selects the point that minimizes the mean of the objective function.

16.2 Error-Based Exploration

Error-based exploration seeks to increase confidence in the true function. A Gaussian process can tell us both the mean and standard deviation at every point. A large standard deviation indicates low confidence, so error-based exploration samples at design points with maximum uncertainty.

The next sample point is:

$$x^{(m+1)} = \underset{\mathbf{x} \in \mathcal{X}}{\arg\max}\, \hat{\sigma}(\mathbf{x}) \tag{16.2}$$

where $\hat{\sigma}(\mathbf{x})$ is the standard deviation of a Gaussian process at a design point $\mathbf{x}$ based on the previous m design points. The process is illustrated in figure 16.2.

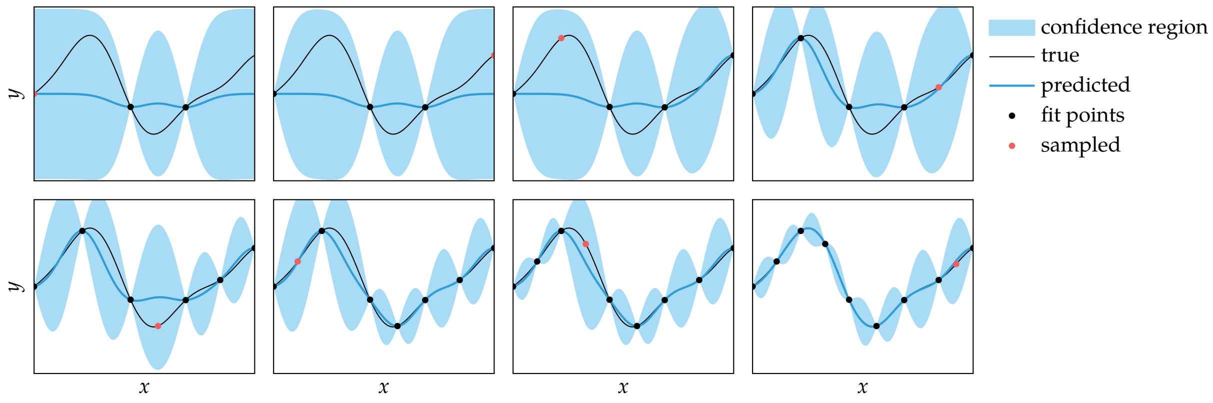


Figure 16.2. Error-based exploration selects a point with maximal uncertainty.

Gaussian processes are often defined over all of $\mathbb{R}^n$. Optimization problems with unbounded feasible sets will always have high uncertainty far away from sampled points, making it impossible to become confident in the true underlying function over the entire domain. Error-based exploration must thus be constrained to a closed region.

16.3 *Lower Confidence Bound Exploration*

While error-based exploration reduces the uncertainty in the objective function overall, its samples are often in regions that are unlikely to contain a global minimum. *Lower confidence bound exploration* trades off between greedy minimization employed by prediction-based optimization and uncertainty reduction employed by error-based exploration. The next sample minimizes the *lower confidence bound* of the objective function

$$LB(\mathbf{x}) = \hat{\mu}(\mathbf{x}) - \alpha\hat{\sigma}(\mathbf{x}) \tag{16.3}$$

where $\alpha \geq 0$ is a constant that controls the trade-off between *exploration* and *exploitation*. Exploration involves minimizing uncertainty, and exploitation involves minimizing the predicted mean. We have prediction-based optimization with $\alpha = 0$, and we have error-based exploration as α approaches ∞. The process is illustrated in figure 16.3.

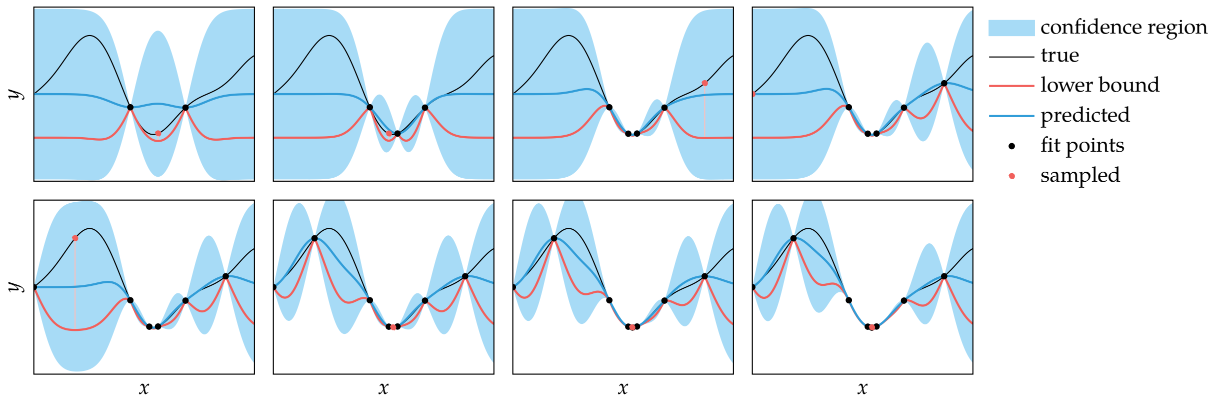


Figure 16.3. Lower confidence bound exploration trades off between minimizing uncertainty and minimizing the predicted function.

16.4 *Probability of Improvement Exploration*

We can sometimes obtain faster convergence by selecting the design point that maximizes the chance that the new point will be better than the samples we have seen so far. The *improvement* for a function sampled at $\mathbf{x}$ producing $y = f(\mathbf{x})$ is

$$I(y) = \begin{cases} y_{\text{min}} - y & \text{if } y < y_{\text{min}} \\ 0 & \text{otherwise} \end{cases} \tag{16.4}$$

where $y_{\min}$ is the minimum value sampled so far.

The *probability of improvement* at points where $\hat{\sigma} > 0$ is

$$P(y < y_{\min}) = \int_{-\infty}^{y_{\min}} \mathcal{N}(y \mid \hat{\mu}, \hat{\sigma}^2)\,dy \tag{16.5}$$

$$= \Phi\left(\frac{y_{\min} - \hat{\mu}}{\hat{\sigma}}\right) \tag{16.6}$$

where Φ is the *standard normal cumulative distribution function* (see appendix C.7). This calculation (algorithm 16.1) is shown in figure 16.4. Figure 16.5 illustrates this process. When $\hat{\sigma} = 0$, which occurs at points where we have noiseless measurements, the probability of improvement is zero.

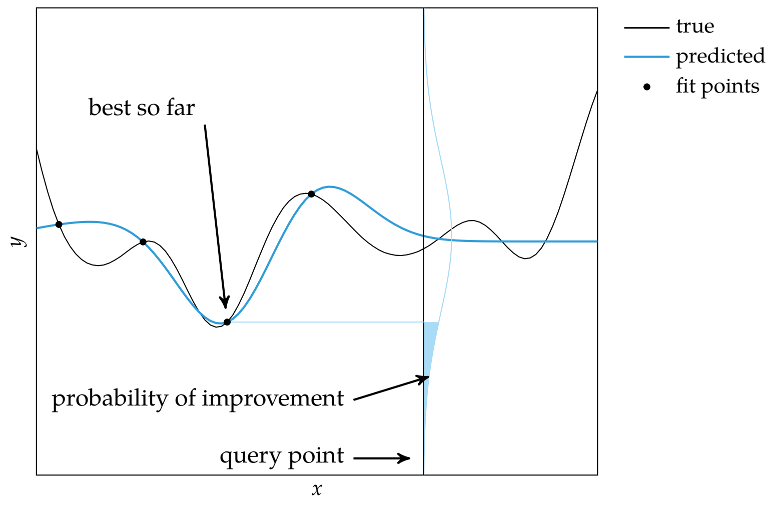


Figure 16.4. The probability of improvement is the probability that evaluating a particular point will yield a better result than the best so far. This figure shows the probability density function predicted at a query point, with the shaded region below $y_{\min}$ corresponding to the probability of improvement.

```
prob_of_improvement(y_min, μ, σ) = cdf(Normal(μ, σ), y_min)
```

Algorithm 16.1. Computing the probability of improvement for a given best y value `y_min`, mean `μ`, and variance `ν`.

16.5 Expected Improvement Exploration

Optimization is concerned with finding the minimum of the objective function. While maximizing the probability of improvement will tend to decrease the objective function over time, it does not improve very much with each iteration.

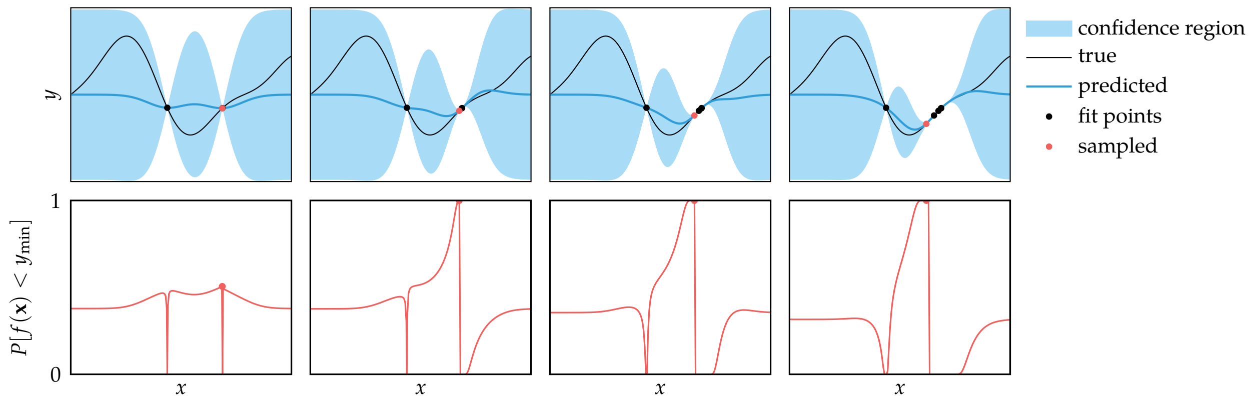


Figure 16.5. Maximizing the probability of improvement selects samples most likely to produce lower objective point values.

We can focus our exploration of points that maximize our *expected improvement* over the current best function value.

Through a substitution

$$z = \frac{y - \hat{\mu}}{\hat{\sigma}} \qquad y'_{\min} = \frac{y_{\min} - \hat{\mu}}{\hat{\sigma}} \tag{16.7}$$

we can write the improvement in equation (16.4) as

$$I(y) = \begin{cases} \hat{\sigma}(y'_{\min} - z) & \text{if } z < y'_{\min} \text{ and } \hat{\sigma} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{16.8}$$

where $\hat{\mu}$ and $\hat{\sigma}$ are the predicted mean and standard deviation at the sample point $\mathbf{x}$.

We can calculate the expected improvement using the distribution predicted by the Gaussian process:

$$\begin{aligned}
\mathbb{E}[I(y)] &= \hat{\sigma} \int_{-\infty}^{y'_{\min}} (y'_{\min} - z)\mathcal{N}(z \mid 0, 1)\, dz && (16.9)\\
&= \hat{\sigma}\left[y'_{\min} \int_{-\infty}^{y'_{\min}} \mathcal{N}(z \mid 0,1)\, dz - \int_{-\infty}^{y'_{\min}} z\,\mathcal{N}(z \mid 0,1)\, dz\right] && (16.10)\\
&= \hat{\sigma}\left[y'_{\min} P(z \le y'_{\min}) + \mathcal{N}(y'_{\min} \mid 0,1) - \underbrace{\mathcal{N}(-\infty \mid 0,1)}_{=0}\right] && (16.11)\\
&= (y_{\min} - \hat{\mu})P(y \le y_{\min}) + \hat{\sigma}^2 \mathcal{N}(y_{\min} \mid \hat{\mu}, \hat{\sigma}^2) && (16.12)
\end{aligned}$$

Figure 16.6 illustrates this process using algorithm 16.2.

```
function expected_improvement(y_min, μ, σ)
    p_imp = prob_of_improvement(y_min, μ, σ)
    p_ymin = pdf(Normal(μ, σ), y_min)
    return (y_min - μ)*p_imp + σ^2*p_ymin
end
```

Algorithm 16.2. Computing the expected improvement for a given best y value `y_min`, mean `μ`, and standard deviation `σ`.

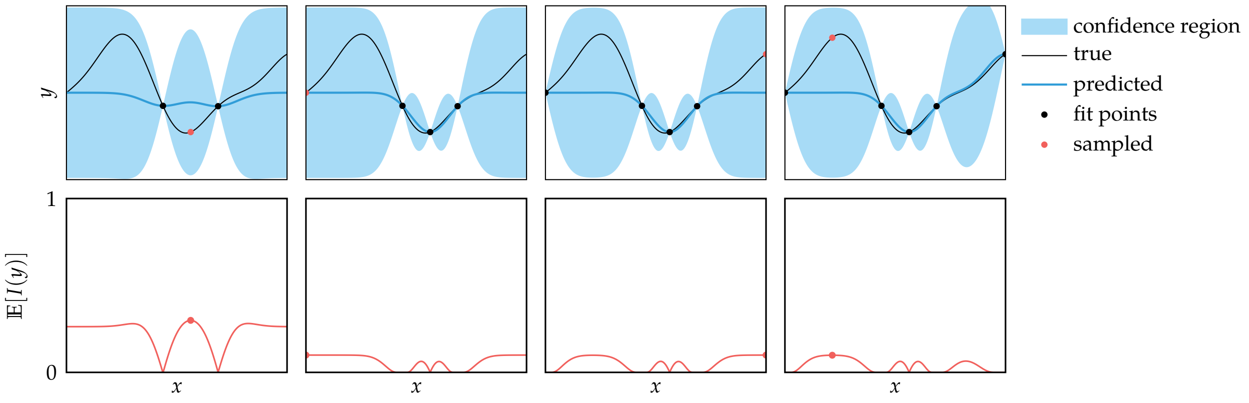


Figure 16.6. Maximizing the expected improvement selects samples which are likely to improve the lower bound by as much as possible.

16.6 *Safe Optimization*

In some contexts, it may be costly to evaluate points that are deemed unsafe, which may correspond to low performing or infeasible points. Problems such as the in-flight tuning of the controller of a drone or safe movie recommendations require *safe exploration*—searching for an optimal design point while carefully avoiding sampling an unsafe design.

This section outlines the *SafeOpt* algorithm,[3] which addresses a class of safe exploration problems. We sample a series of design points $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}$ in pursuit of a minimum but without $f(\mathbf{x}^{(i)})$ exceeding a critical safety threshold $y_{\max}$. In addition, we receive only noisy measurements of the objective function, where the noise is zero-mean with variance ν. Such an objective function and its associated safe regions are shown in figure 16.7.

The SafeOpt algorithm uses Gaussian process surrogate models for prediction. At each iteration, we fit a Gaussian process to the noisy samples from f. After the

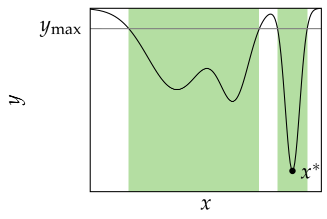


Figure 16.7. SafeOpt solves safe exploration problems that minimize f while remaining within safe regions defined by maximum objective function values.

[3] Y. Sui, A. Gotovos, J. Burdick, and A. Krause, "Safe Exploration for Optimization with Gaussian Processes," in *International Conference on Machine Learning* (ICML), vol. 37, 2015.

ith sample, SafeOpt calculates the upper and lower confidence bounds:

$$u_i(\mathbf{x}) = \hat{\mu}_{i-1}(\mathbf{x}) + \sqrt{\beta \hat{\nu}_{i-1}(\mathbf{x})} \tag{16.13}$$

$$\ell_i(\mathbf{x}) = \hat{\mu}_{i-1}(\mathbf{x}) - \sqrt{\beta \hat{\nu}_{i-1}(\mathbf{x})} \tag{16.14}$$

where larger values of β yield wider confidence regions. Such bounds are shown in figure 16.8.

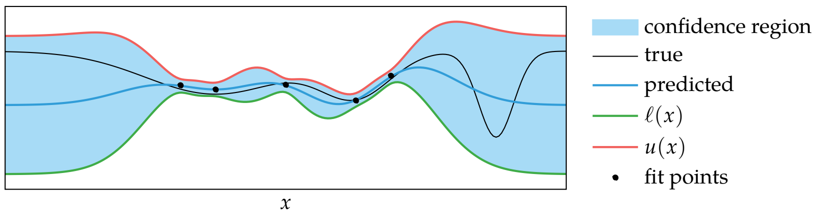


Figure 16.8. An illustration of functions based on the predictions of a Gaussian process used by SafeOpt.

The Gaussian process predicts a distribution over $f(\mathbf{x})$ for any design point. Being Gaussian, these predictions can provide only a probabilistic guarantee of safety up to an arbitrary factor:[4]

[4] Note the similarity to the probability of improvement.

$$P(f(\mathbf{x}) \leq y_{\max}) = \Phi\left(\frac{y_{\max} - \hat{\mu}(\mathbf{x})}{\sqrt{\hat{\nu}(\mathbf{x})}}\right) \geq P_{\text{safe}} \tag{16.15}$$

The predicted safe region $\mathcal{S}$ consists of the design points that provide a probability of safety greater than the required level P_{safe}, as illustrated in figure 16.9. The safe region can also be defined in terms of Lipschitz upper bounds constructed from upper bounds evaluated at previously sampled points.

SafeOpt chooses a safe sample point that balances the desire to localize a reachable minimizer of f and to expand the safe region. The set of potential minimizers of f is denoted $\mathcal{M}$ (figure 16.10), and the set of points that will potentially lead to the expansion of the safe regions is denoted $\mathcal{E}$ (figure 16.11). To trade off exploration and exploitation, we choose the design point $\mathbf{x}$ with the largest predictive variance among both sets $\mathcal{M}$ and $\mathcal{E}$.[5]

[5] For a variation of this algorithm, see F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe Controller Optimization for Quadrotors with Gaussian Processes," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

The set of potential minimizers consists of the safe points whose lower confidence bound is lower than the lowest upper bound:

$$\mathcal{M}_i = \left\{ \mathbf{x} \in \mathcal{S}_i \mid \ell_i(\mathbf{x}) \leq \min_{\mathbf{x}' \in \mathcal{S}_i} u_i(\mathbf{x}') \right\} \tag{16.16}$$

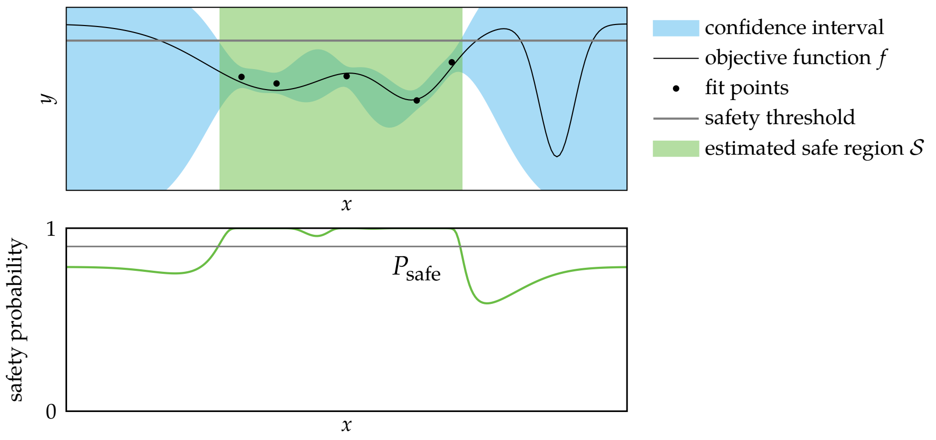


Figure 16.9. The safety regions (green) predicted by a Gaussian process.

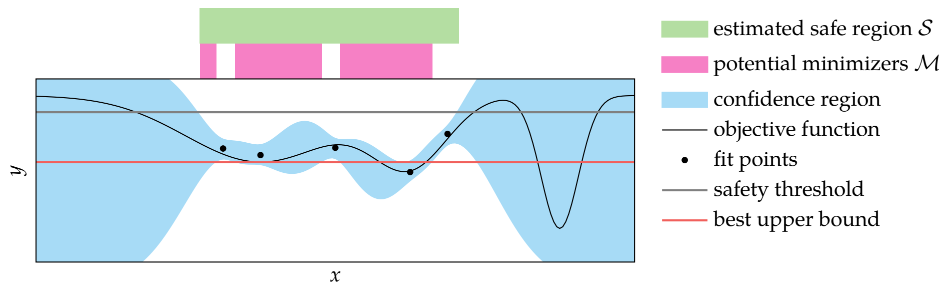


Figure 16.10. The potential minimizers are the safe points whose lower bounds are lower than the best, safe upper bound.

At step i, the set of potential expanders $\mathcal{E}_i$ consists of the safe points that, if added to the Gaussian process, optimistically assuming the lower bound, produce a posterior distribution with a larger safe set. The potential expanders naturally lie near the boundary of the safe region.

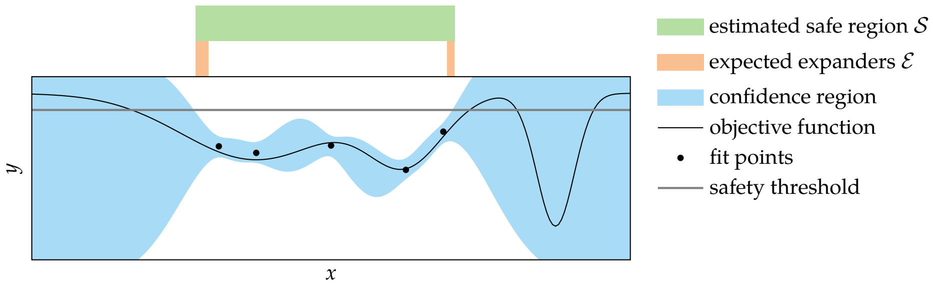


Figure 16.11. The set of potential expanders.

Given an initial safe point[6] $\mathbf{x}^{(1)}$, SafeOpt chooses the design point among sets $\mathcal{M}$ and $\mathcal{E}$ with the greatest uncertainty, as quantified by the width $w_i(x) = u(x) - \ell(x)$:

$$x^{(i)} = \underset{\mathbf{x}\in\mathcal{M}_i\cup\mathcal{E}_i}{\arg\max}\, w_i(\mathbf{x}) \tag{16.17}$$

[6] SafeOpt cannot guarantee safety if it is not initialized with at least one point that it knows is safe.

SafeOpt proceeds until a termination condition is met. It is common to run the algorithm for a fixed number of iterations or until the maximum width is less than a set threshold.

Maintaining sets in multidimensional spaces can be computationally challenging. SafeOpt assumes a finite design space $\mathcal{X}$ that can be obtained with a sampling method applied over the continuous search domain. Increasing the density of the finite design space leads to more accurate results with respect to the continuous space, but it takes longer per iteration.

SafeOpt is implemented in algorithm 16.3, and calls algorithm 16.4 to update the predicted confidence intervals; algorithm 16.5 to compute the safe, minimizer, and expander regions; and algorithm 16.6 to select a query point. The progression of SafeOpt is shown for one dimension in figure 16.12, and for two dimensions in figure 16.13.

```
function safe_opt(GP, X, i, f, y_max; β=3.0, k_max=10)
    push!(GP, X[i], f(X[i])) # make first observation

    m = length(X)
    u, l = fill(Inf, m), fill(-Inf, m)
    S, M, E = falses(m), falses(m), falses(m)

    for k in 1 : k_max
        update_confidence_intervals!(GP, X, u, l, β)
        compute_sets!(GP, S, M, E, X, u, l, y_max, β)
        i = get_new_query_point(M, E, u, l)
        i != 0 || break
        push!(GP, X[i], f(X[i]))
    end

    # return the best point
    update_confidence_intervals!(GP, X, u, l, β)
    S[:] = u .≤ y_max
    if any(S)
        u_best, i_best = findmin(u[S])
        i_best = findfirst(isequal(i_best), cumsum(S))
        return (u_best, i_best)
    else
        return (NaN,0)
    end
end
```

Algorithm 16.3. The SafeOpt algorithm applied to an empty Gaussian process `GP`, a finite design space `X`, index of initial safe point `i`, objective function `f`, and safety threshold `y_max`. The optional parameters are the confidence scalar `β` and the number of iterations `k_max`. A tuple containing the best safe upper bound and its index in `X` is returned.

```
function update_confidence_intervals!(GP, X, u, l, β)
    μₚ, νₚ = predict(GP, X)
    u[:] = μₚ + sqrt.(β*νₚ)
    l[:] = μₚ - sqrt.(β*νₚ)
    return (u, l)
end
```

Algorithm 16.4. A method for updating the lower and upper bounds used in SafeOpt, which takes the Gaussian process `GP`, the finite search space `X`, the upper and lower-bound vectors `u` and `l`, and the confidence scalar `β`.

```
function compute_sets!(GP, S, M, E, X, u, l, y_max, β)
    fill!(M, false)
    fill!(E, false)

    # safe set
    S[:] = u .≤ y_max

    if any(S)

        # potential minimizers
        M[S] = l[S] .< minimum(u[S])

        # maximum width (in M)
        w_max = maximum(u[M] - l[M])

        # expanders - skip values in M or those with w ≤ w_max
        E[:] = S .& .~M # skip points in M
        if any(E)
            E[E] .= maximum(u[E] - l[E]) .> w_max
            for (i,e) in enumerate(E)
                if e && u[i] - l[i] > w_max
                    push!(GP, X[i], l[i])
                    μₚ, νₚ = predict(GP, X[.~S])
                    pop!(GP)
                    E[i] = any(μₚ + sqrt.(β*νₚ) .≥ y_max)
                    if E[i]; w_max = u[i] - l[i]; end
                end
            end
        end
    end

    return (S,M,E)
end
```

Algorithm 16.5. A method for updating the safe `S`, minimizer `M`, and expander `E` sets used in SafeOpt. The sets are all Boolean vectors indicating whether the corresponding design point in `X` is in the set. The method also takes the Gaussian process `GP`, the upper and lower bounds `u` and `l`, respectively, the safety threshold `y_max`, and the confidence scalar `β`.

```
function get_new_query_point(M, E, u, l)
    ME = M .| E
    if any(ME)
            v = argmax(u[ME] - l[ME])
        return findfirst(isequal(v), cumsum(ME))
    else
        return 0
    end
end
```

Algorithm 16.6. A method for obtaining the next query point in SafeOpt. The index of the point in `X` with the greatest width is returned.

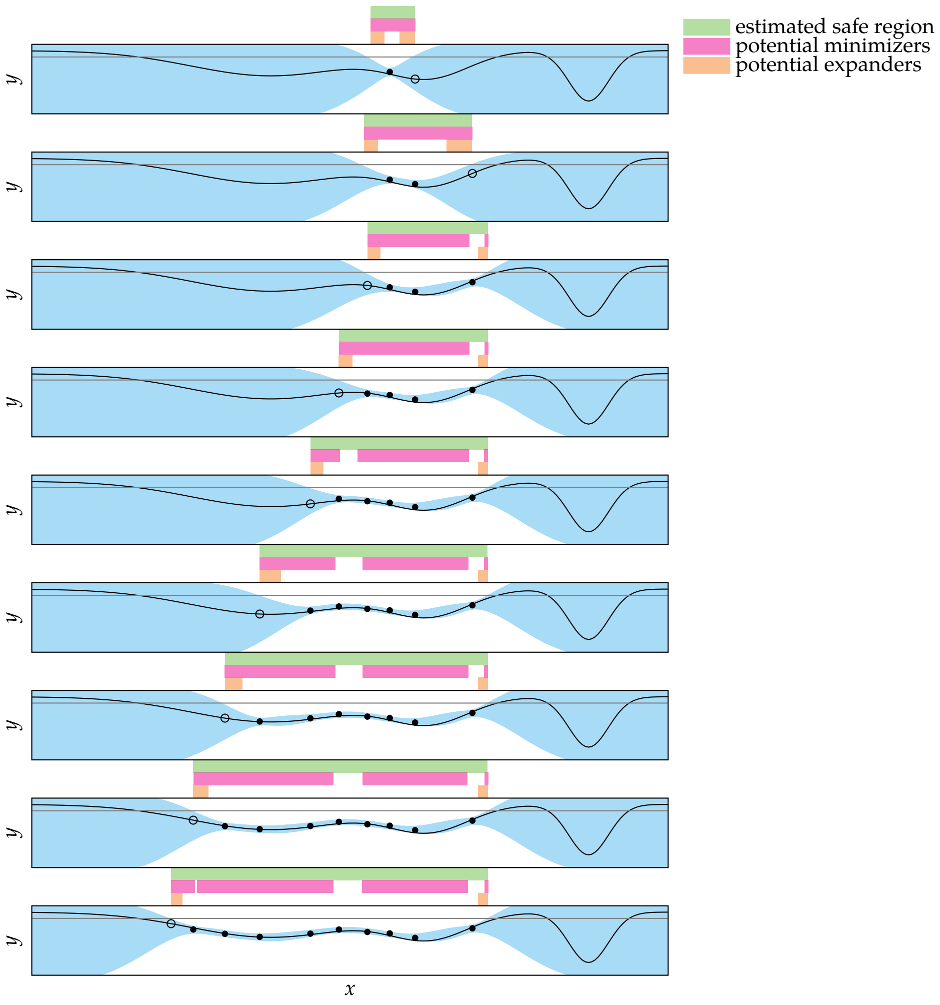


Figure 16.12. The first eight iterations of SafeOpt on a univariate function. SafeOpt can never reach the global optimum on the right-hand side because it requires crossing an unsafe region. We can only hope to find the global minima in our locally reachable safe region.

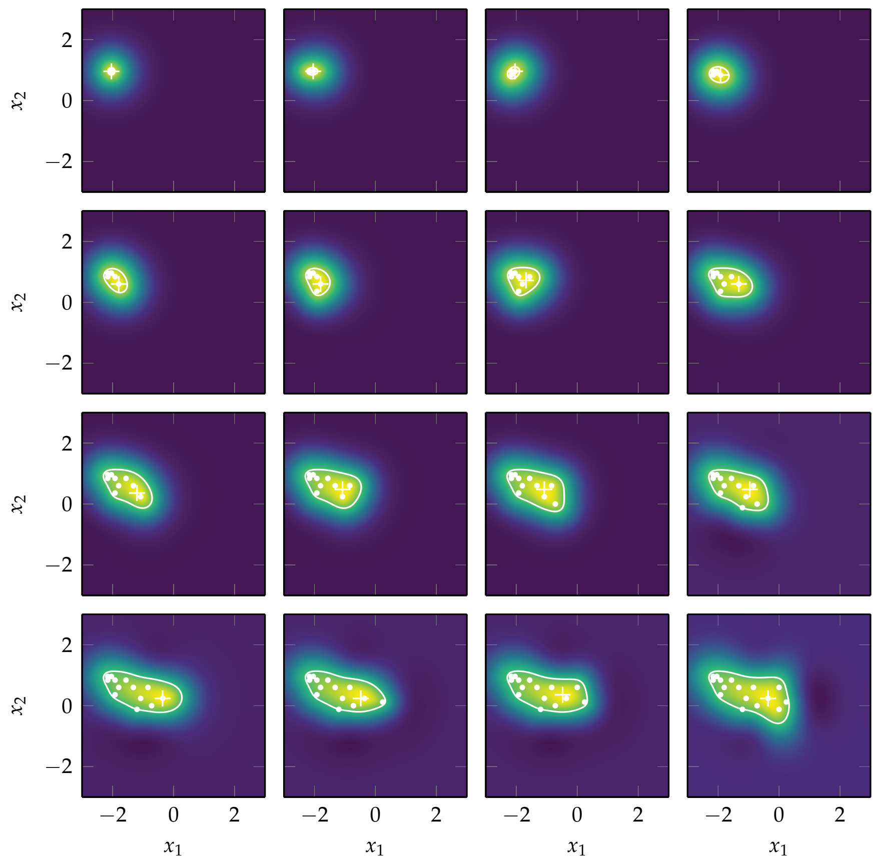
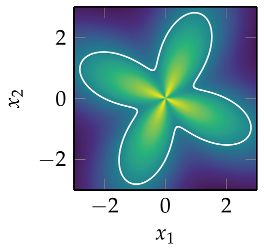


Figure 16.13. SafeOpt applied to the flower function (appendix B.4) with $y_{\max} = 2$, a Gaussian process mean of $\mu(\mathbf{x}) = 2.5$, variance $\nu = 0.01$, $\beta = 10$, a 51×51 uniform grid over the search space, and an initial point $x^{(1)} = [-2.04, 0.96]$. The color indicates the value of the upper bound, the cross indicates the safe point with the lowest upper bound, and the white contour line is the estimated safe region.

The objective function with the true safe region outlined in white:



16.7 Summary

- Gaussian processes can be used to guide the optimization process using a variety of strategies that use estimates of quantities such as the lower confidence bound, probability of improvement, and expected improvement.
- Some problems do not allow for the evaluation of unsafe designs, in which case we can use safe exploration strategies that rely on Gaussian processes.

16.8 Exercises

Exercise 16.1. Give an example in which prediction-based optimization fails.

Exercise 16.2. What is the main difference between lower confidence bound exploration and error-based exploration in the context of optimization?

Exercise 16.3. We have a function $f(x) = (x-2)^2/40 - 0.5$ with $x \in [-5, 5]$, and we have evaluation points at -1 and 1. Assume we use a Gaussian process surrogate model with a zero-mean function, and a squared exponential kernel $\exp(-r^2/2)$, where r is the Euclidean distance between two points. Which value for x would we evaluate next if we were maximizing probability of improvement? Which value for x would we evaluate next if we were maximizing expected improvement?