Surrogate Optimization: Expected Improvement

BIOE 498/598 PJ

Spring 2021

Exploration vs. exploitation

There is a fundamental tradeoff in global optimization:

- **Exploration** searches areas of high uncertainty to find *new* regions of interest.
- **Exploitation** refines existing optima by adding points to *known* regions of interest.

Exploration vs. exploitation

There is a fundamental tradeoff in global optimization:

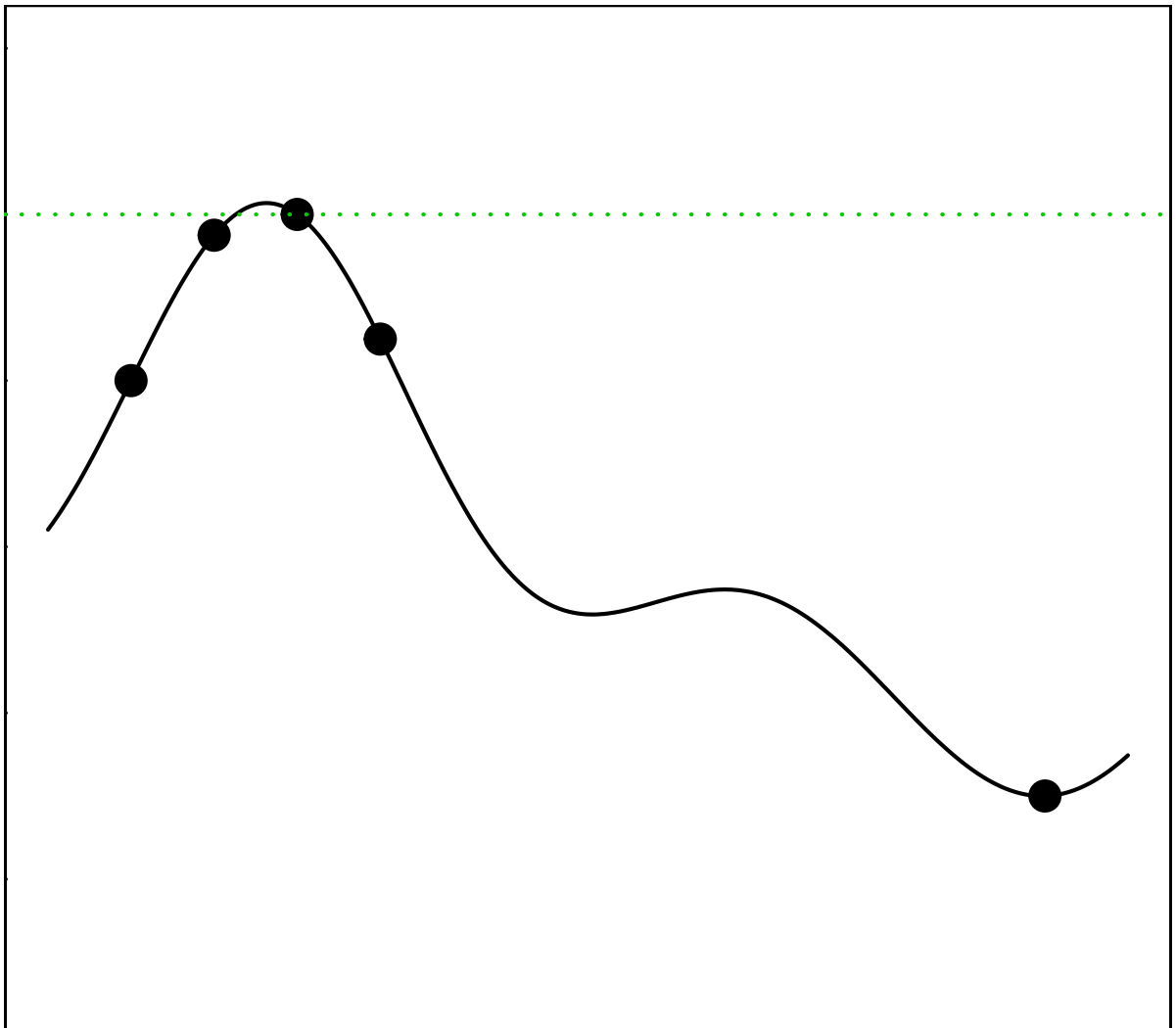
- **Exploration** searches areas of high uncertainty to find *new* regions of interest.
- **Exploitation** refines existing optima by adding points to *known* regions of interest.

Should we explore or exploit?

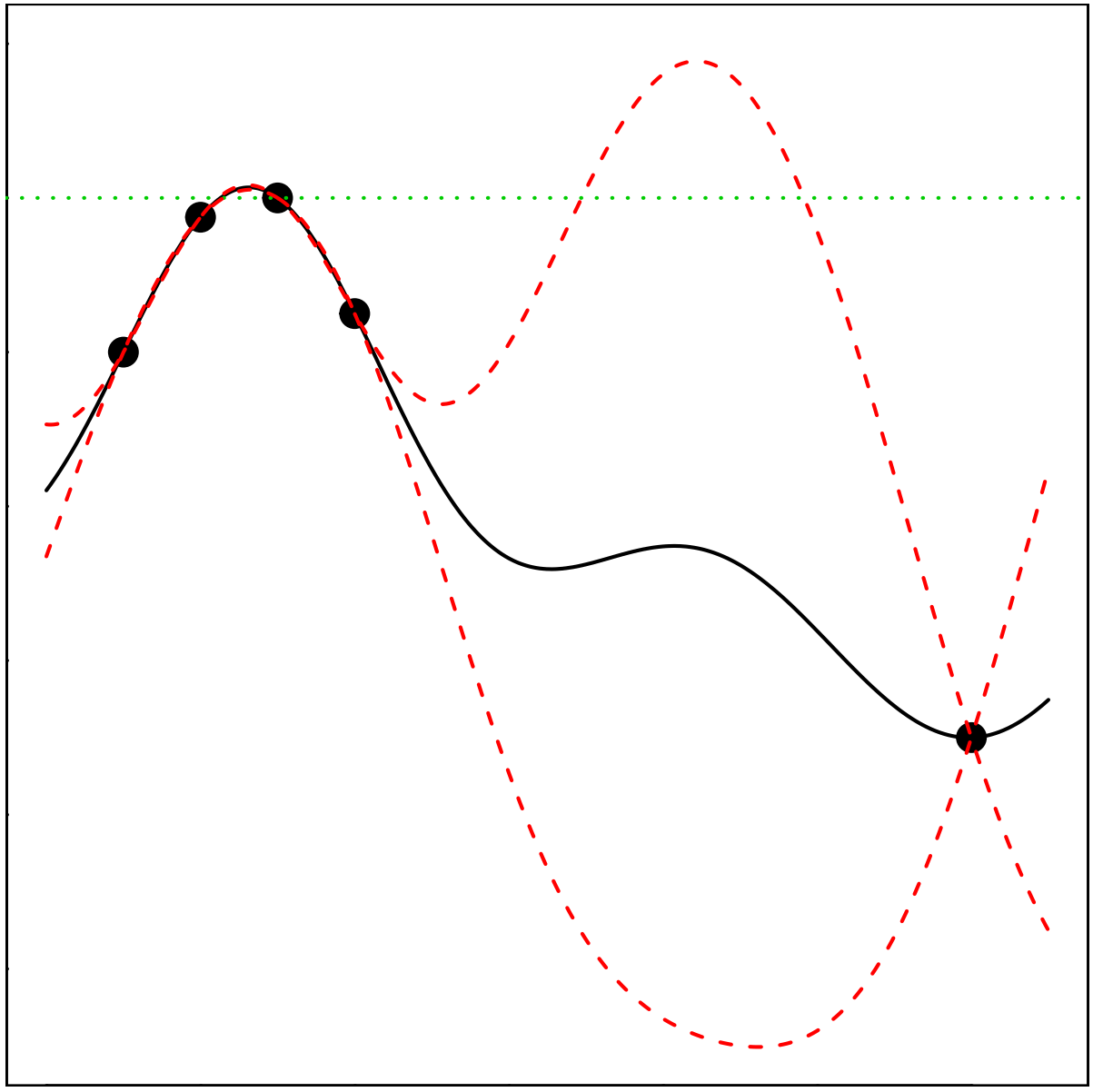
- **Both.** Good algorithms balance discovery and refinement.
- The *best* balance is an open problem. Some solutions:
  - Always explore some (small) percent of the time.
  - Explore early, exploit later.
  - Alternate between batches of exploration and exploitation.
  - **Today:** Combine exploration and exploitation into a single metric.

A 1-D example (Gramacy 2020)

```
Xn <- c(1, 2, 3, 4, 12)
yn <- c(0, 1.75, 2, 0.5, -5)
gp <- newGP(matrix(Xn, ncol=1), yn, d=10, g=1e-8)
X <- seq(0, 13, length=1000)
p <- predGP(gp, matrix(X, ncol=1), lite=TRUE)
```



What happens when we consider uncertainty?



Optimizing for objective improvement

A key insight in Bayesian optimization was the switch to *expected improvement* (Schonlau 1997).

As usual, assume we've measured n responses y_n at locations X_n. Define

$$y_{\max} = \max\{y_1, \ldots, y_n\}.$$

Optimizing for objective improvement

A key insight in Bayesian optimization was the switch to *expected improvement* (Schonlau 1997).

As usual, assume we've measured n responses y_n at locations X_n. Define

$$y_{\max} = \max\{y_1, \ldots, y_n\}.$$

The *improvement* in the objective at a new input x is

$$I(x) = \max\{0, y(x) - y_{\max}\}$$

where the maximization "floors" the improvement at zero.

Optimizing for objective improvement

A key insight in Bayesian optimization was the switch to *expected improvement* (Schonlau 1997).

As usual, assume we've measured n responses y_n at locations X_n. Define

$$y_{\max} = \max\{y_1, \ldots, y_n\}.$$

The *improvement* in the objective at a new input x is

$$I(x) = \max\{0, y(x) - y_{\max}\}$$

where the maximization "floors" the improvement at zero.

The *expected improvement* $\mathrm{EI}(x) = \mathbb{E}\{I(x)\}$ quantifies how much we expect the best objective value to increase after measuring at point x.

Expected Improvement

The model's predictions $y(x)$ are stochastic. How do we estimate the expected improvement

$$\mathrm{EI}(x) = \mathbb{E}\{I(x)\} = \mathbb{E}\{\max[0, y(x) - y_{\max}]\}?$$

Expected Improvement

The model's predictions $y(x)$ are stochastic. How do we estimate the expected improvement

$$\mathrm{EI}(x) = \mathbb{E}\{I(x)\} = \mathbb{E}\{\max[0, y(x) - y_{\max}]\}?$$

We can sample $y(x)$ many times, averaging the improvement $I(x)$ for T samples:

$$\mathrm{EI}(x) \approx \frac{1}{T} \sum_{i=1}^{T} \max\{0, y_i(x) - y_{\max}\}.$$

Expected Improvement

The model's predictions $y(x)$ are stochastic. How do we estimate the expected improvement

$$\mathrm{EI}(x) = \mathbb{E}\{I(x)\} = \mathbb{E}\{\max[0, y(x) - y_{\max}]\}?$$

We can sample $y(x)$ many times, averaging the improvement $I(x)$ for T samples:

$$\mathrm{EI}(x) \approx \frac{1}{T} \sum_{i=1}^{T} \max\{0, y_i(x) - y_{\max}\}.$$

Even better, we can leverage that GPR predictions are multivariate normal with mean $\mu(x)$ and variance $\sigma(x)$. Let $z = (\mu(x) - y_{\max})/\sigma(x)$. Then

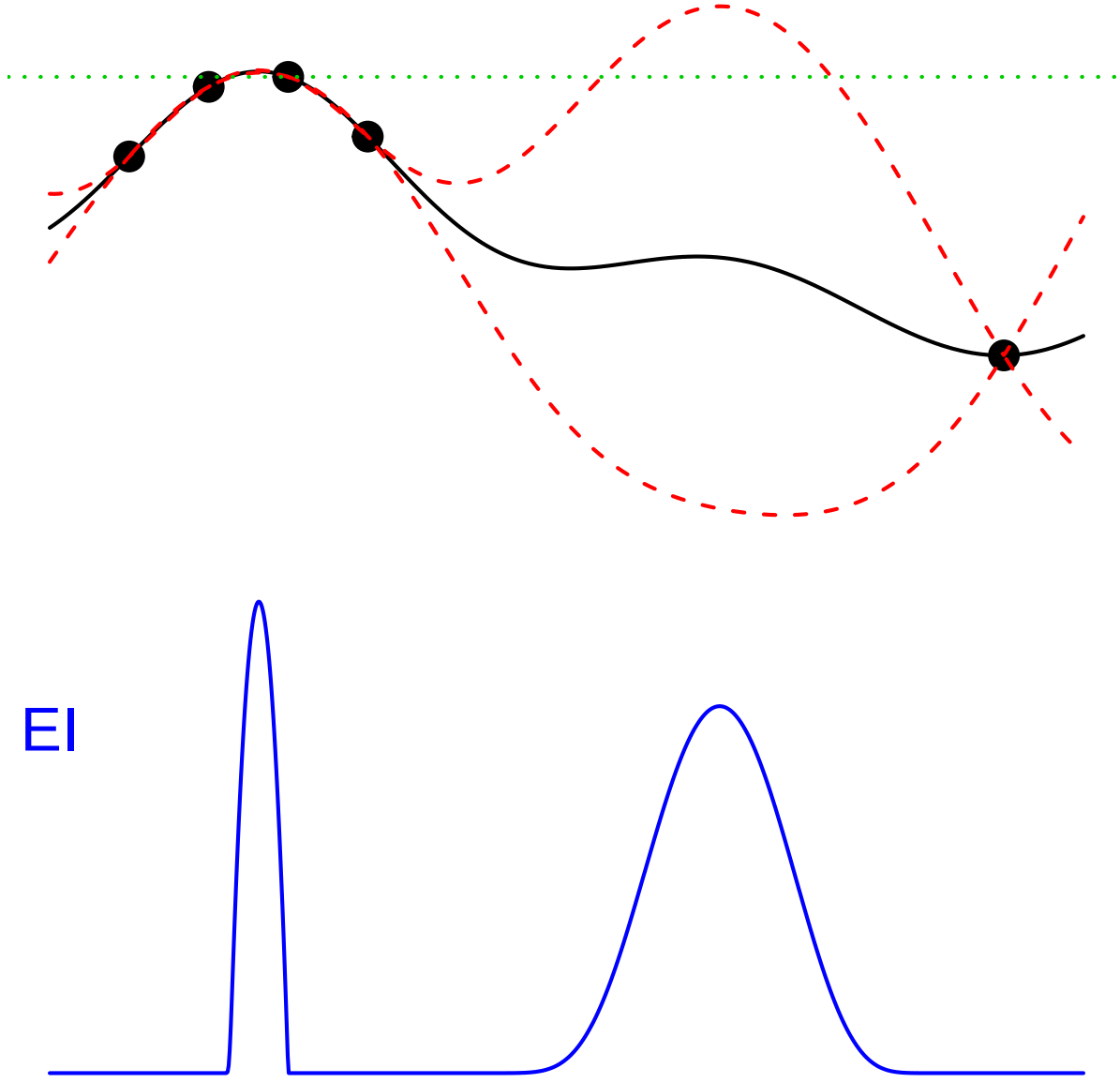
$$\mathrm{EI}(x) = (\mu(x) - y_{\max})\mathrm{CDF}(z) + \sigma(x)PDF(z)$$

using the PDF and CDF of a standard Gaussian distribution.

Calculating EI

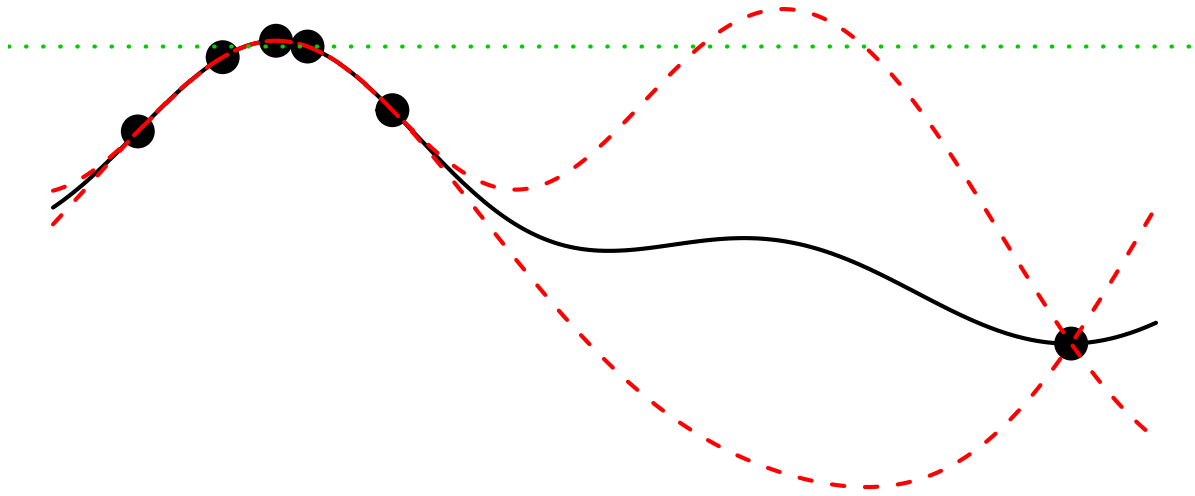
```
argmax <- which.max(yn)
ymax <- yn[argmax]
z <- (p$mean - ymax)/sqrt(p$s2)
ei <- (p$mean - ymax)*pnorm(z) + sqrt(p$s2)*dnorm(z)
```

Calculating EI



Picking the next sample x

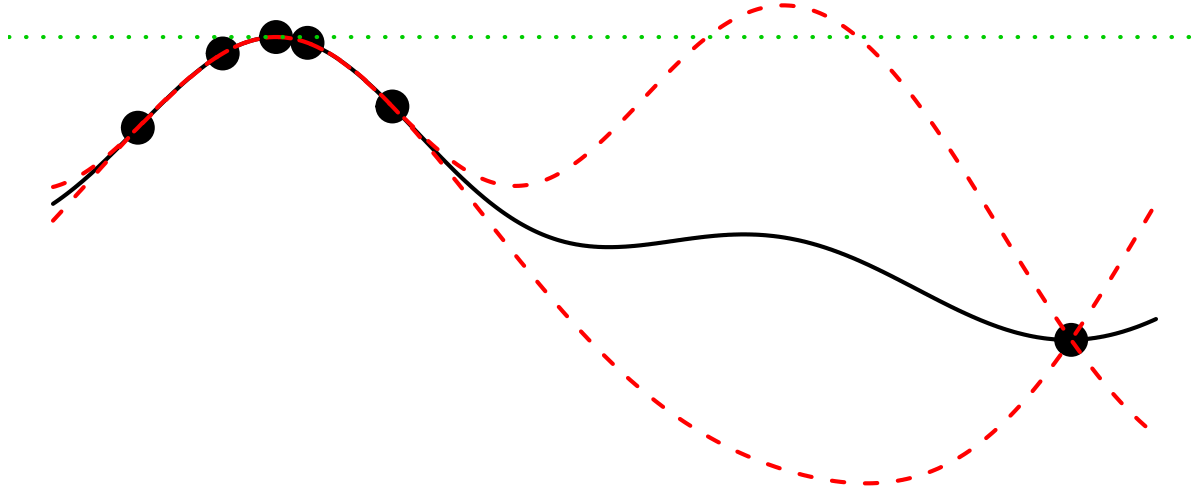
```
argmaxEI <- which.max(ei)
Xn <- c(Xn, X[argmaxEI])
yn <- c(yn, p$mean[argmaxEI])

updateGP(gp, matrix(X[argmaxEI], ncol=1), p$mean[argmaxEI])
p <- predGP(gp, matrix(X, ncol=1), lite=TRUE)
```



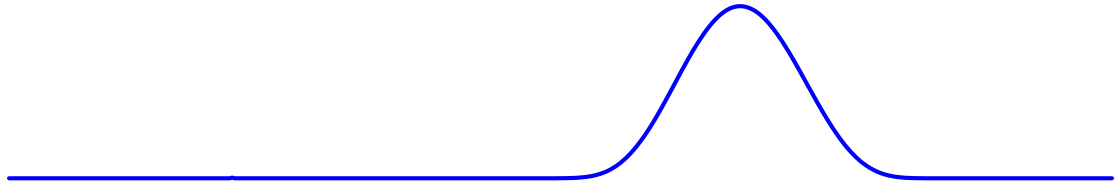
Calculating EI (round 2)

```
argmax <- which.max(yn)
ymax <- yn[argmax]
z <- (p$mean - ymax)/sqrt(p$s2)
ei <- (p$mean - ymax)*pnorm(z) + sqrt(p$s2)*dnorm(z)
```

Calculating EI (round 2)



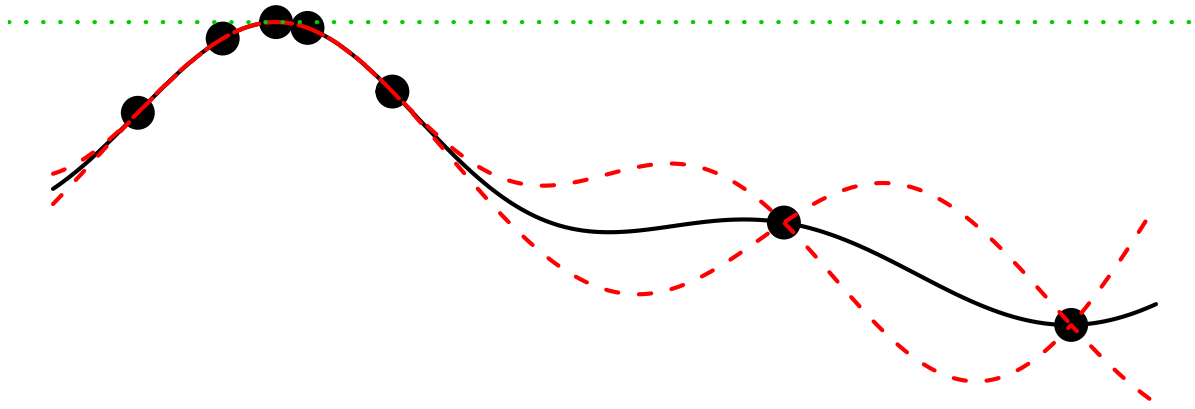
EI



Picking the next sample x (round 2)

```
argmaxEI <- which.max(ei)
Xn <- c(Xn, X[argmaxEI])
yn <- c(yn, p$mean[argmaxEI])

updateGP(gp, matrix(X[argmaxEI], ncol=1), p$mean[argmaxEI])
p <- predGP(gp, matrix(X, ncol=1), lite=TRUE)
```

After the second update



EI

The complete GPR surrogate optimization framework

To maximize the response y of an unknown function f using no more than N function evaluations:

1. Create a space-filling design X_n for $n < N$.
2. Measure the responses $y_n(X_n)$ and train $\mathcal{GP}(X_n, y_n)$.
3. Use a nonlinear optimizer (`optim`) to find the argmax x of a metric (mean, SD, EI).
4. Measure $y(x)$ and update $\mathcal{GP}(X_{n+1}, y_{n+1})$.
5. Go to #3 and repeat until all N runs are used.
6. Search $\mathcal{GP}(X_N, y_N)$ for the global maximum $y^*(x^*)$.

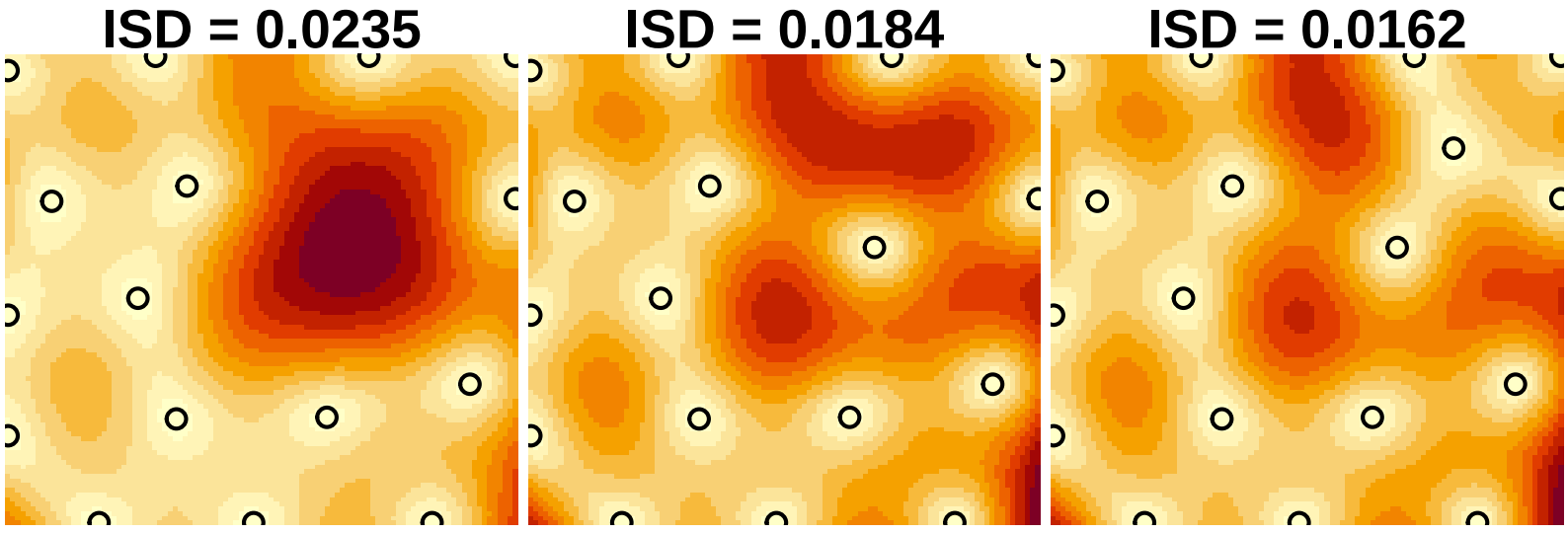
Does sequential design always work?

- Sequential design methods are **last sample optimal**.
- After $N-1$ runs, sequential design finds the optimal location for the last run.

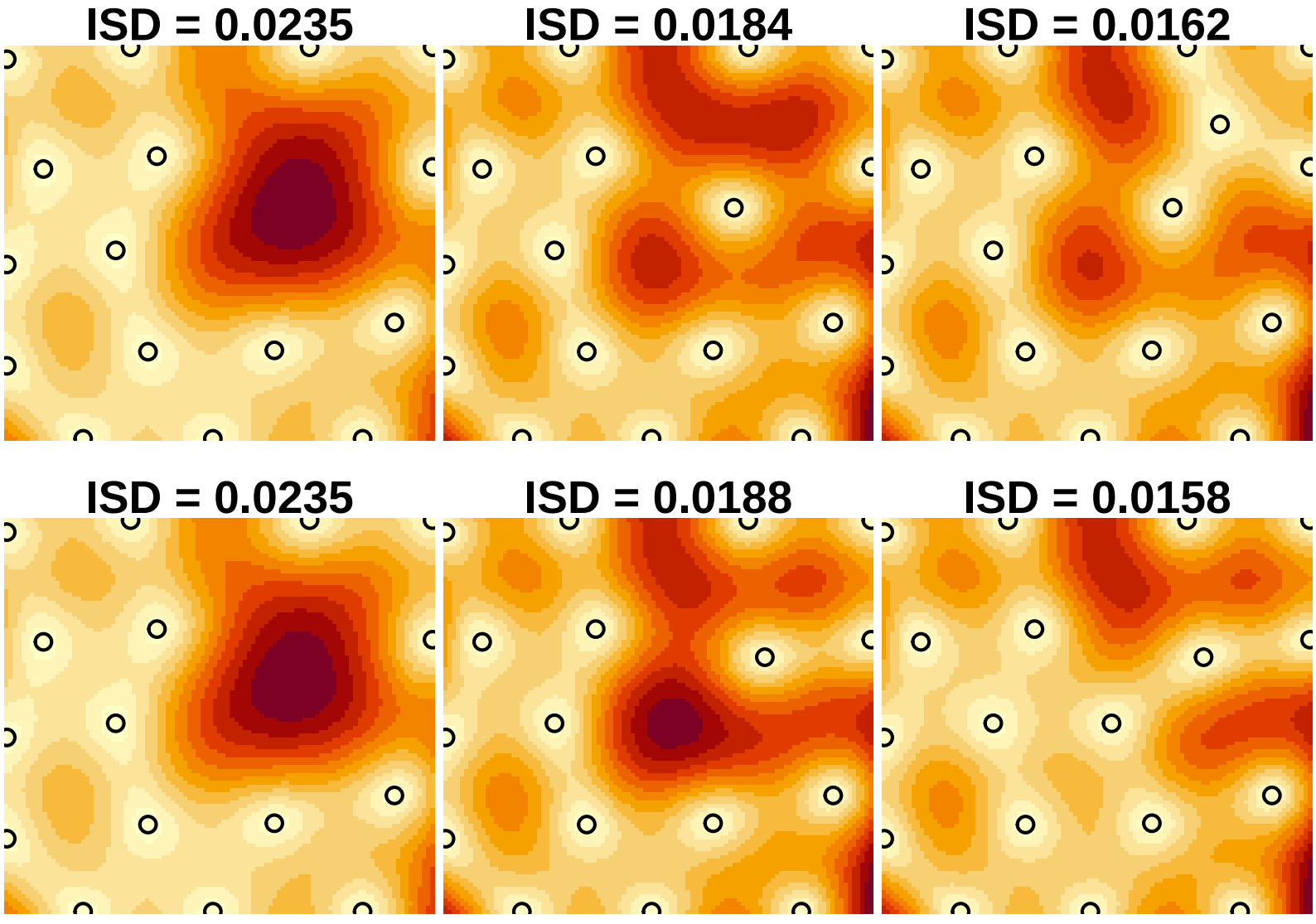
Does sequential design always work?

- Sequential design methods are **last sample optimal**.
- After $N-1$ runs, sequential design finds the optimal location for the last run.
- However, sequential design is *greedy*. If $N-2$ of N runs are finished, two rounds of sequential design may not be optimal.

Limited lookahead in active learning



Limited lookahead in active learning



What's wrong with being greedy?

Imagine we have two runs left. There are two strategies:

1. Select both points with our current information $\mathcal{GP}(X_{N-2}, y_{N-2})$. This ignores the new information available in $\mathcal{GP}(X_{N-1}, y_{N-1})$.
2. Select the first point using current information and select the second point using $\mathcal{GP}(X_{N-1}, y_{N-1})$. The first point ignores the existence of the second point.

What's wrong with being greedy?

Imagine we have two runs left. There are two strategies:

1. Select both points with our current information $\mathcal{GP}(X_{N-2}, y_{N-2})$. This ignores the new information available in $\mathcal{GP}(X_{N-1}, y_{N-1})$.
2. Select the first point using current information and select the second point using $\mathcal{GP}(X_{N-1}, y_{N-1})$. The first point ignores the existence of the second point.

The "best" solution is often a compromise between two extremes. Given a budget of N runs and an initial design X_n, we could

1. Place the remaining $N - n$ runs at once using $\mathcal{GP}(X_n, y_n)$.
2. Place the remaining $N - n$ runs one at a time.

What's wrong with being greedy?

Imagine we have two runs left. There are two strategies:

1. Select both points with our current information $\mathcal{GP}(X_{N-2}, y_{N-2})$. This ignores the new information available in $\mathcal{GP}(X_{N-1}, y_{N-1})$.
2. Select the first point using current information and select the second point using $\mathcal{GP}(X_{N-1}, y_{N-1})$. The first point ignores the existence of the second point.

The "best" solution is often a compromise between two extremes. Given a budget of N runs and an initial design X_n, we could

1. Place the remaining $N - n$ runs at once using $\mathcal{GP}(X_n, y_n)$.
2. Place the remaining $N - n$ runs one at a time.

For example, Let $N = 36$ and $n = 16$, so we have 20 runs to go. We could

1. Place runs in 5 batches of 4 points, **or**
2. Place 4 batches of 4 points, followed by 4 one-at-a-time updates.

Summary

- Surrogate optimization with Gaussian processes finds **global** optima for unknown, expensive functions.
- Balancing *exploration* and *exploitation* is critical for finding the best response.
- Sequential design works well but suffers from limited lookahead.

Summary

- Surrogate optimization with Gaussian processes finds **global** optima for unknown, expensive functions.
- Balancing *exploration* and *exploitation* is critical for finding the best response.
- Sequential design works well but suffers from limited lookahead.

- **Next time:** How do we optimize when function evaluations are *inexpensive*, but the problem is enormous?