Knowledge-Gradient Methods for Bayesian Optimization

Peter I. Frazier
Cornell University
Uber


Jian Wu


Andrew Wilson


Matthias Poloczek


Jialei Wang

Wu, Poloczek, Wilson & F., NIPS'17
Bayesian Optimization with Gradients

Poloczek, Wang & F., NIPS'17
Multi Information-Source Optimization

Wu & F., NIPS'16
The Parallel KG Method for Batch Bayesian Optimization

Bayesian Optimization algorithms use acquisition functions

Generic BO algorithm:

Elicit a prior distribution on the function f (typically a GP).

while (budget is not exhausted) {

Find x that maximizes Acquisition(x,posterior)

Sample at x

Update the posterior distribution

}

There are a few commonly used acquisition functions

Most people use this one

- **Expected Improvement (EI)**

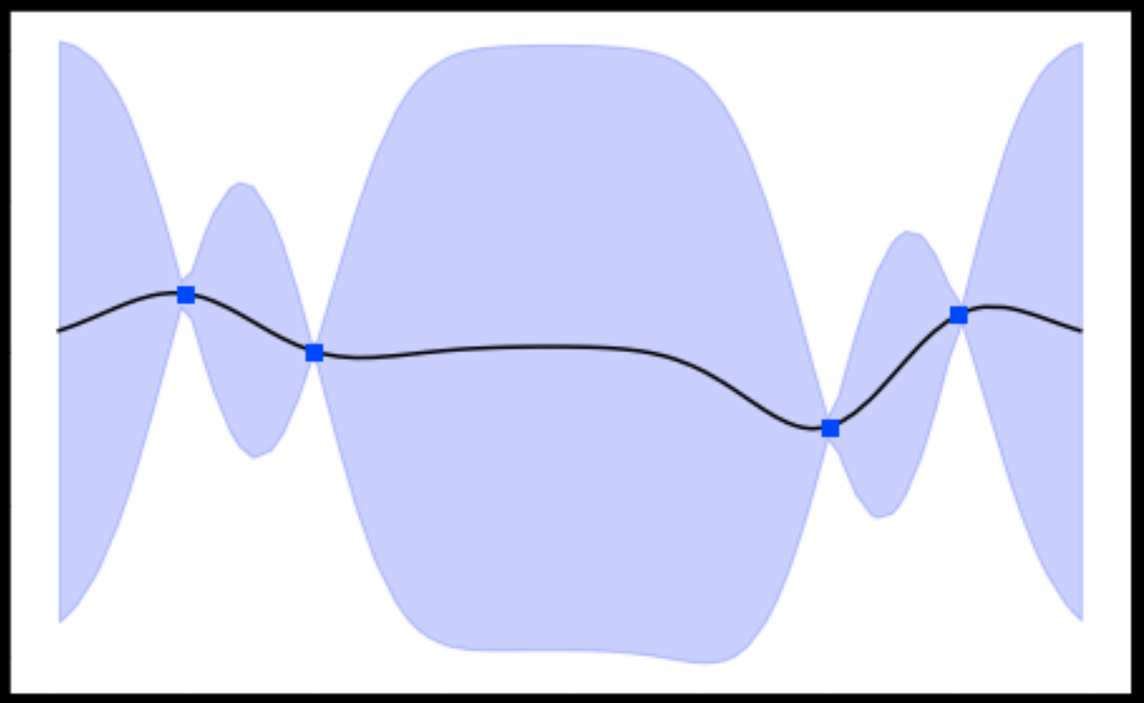
Some people use one of these

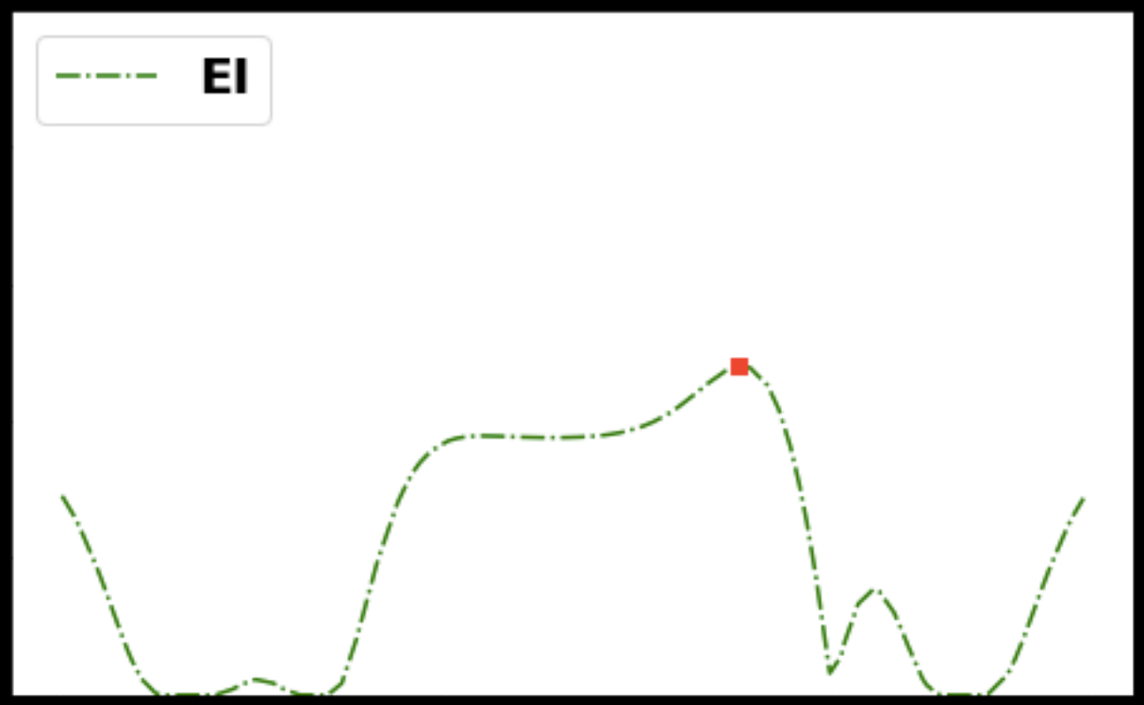
- Expected Improvement (EI)
- **Predictive Entropy Search (PES)**
- **Upper Confidence Bound (UCB)**
- **Probability of Improvement (P*)**

I'll tell you why you might want to try this one, & give you a link to code

- Expected Improvement (EI)
- Upper Confidence Bound (UCB)
- Predictive Entropy Search (PES)
- Probability of Improvement (P*)
- **Knowledge Gradient (KG)**

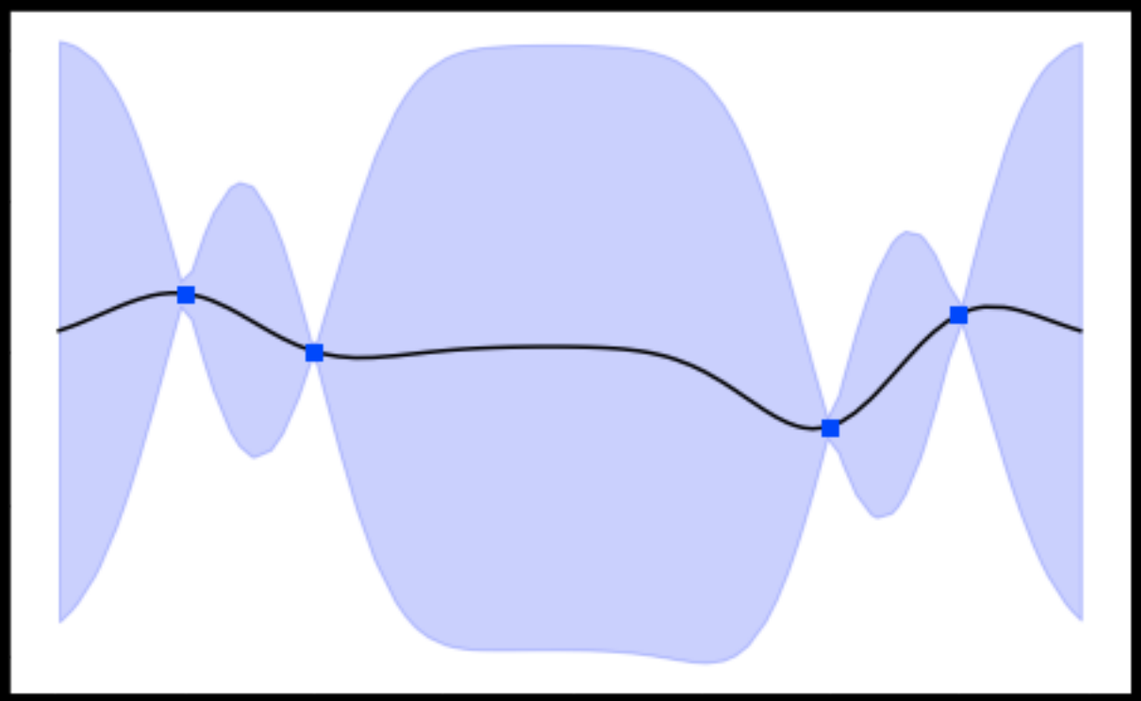
Let me remind you about EI

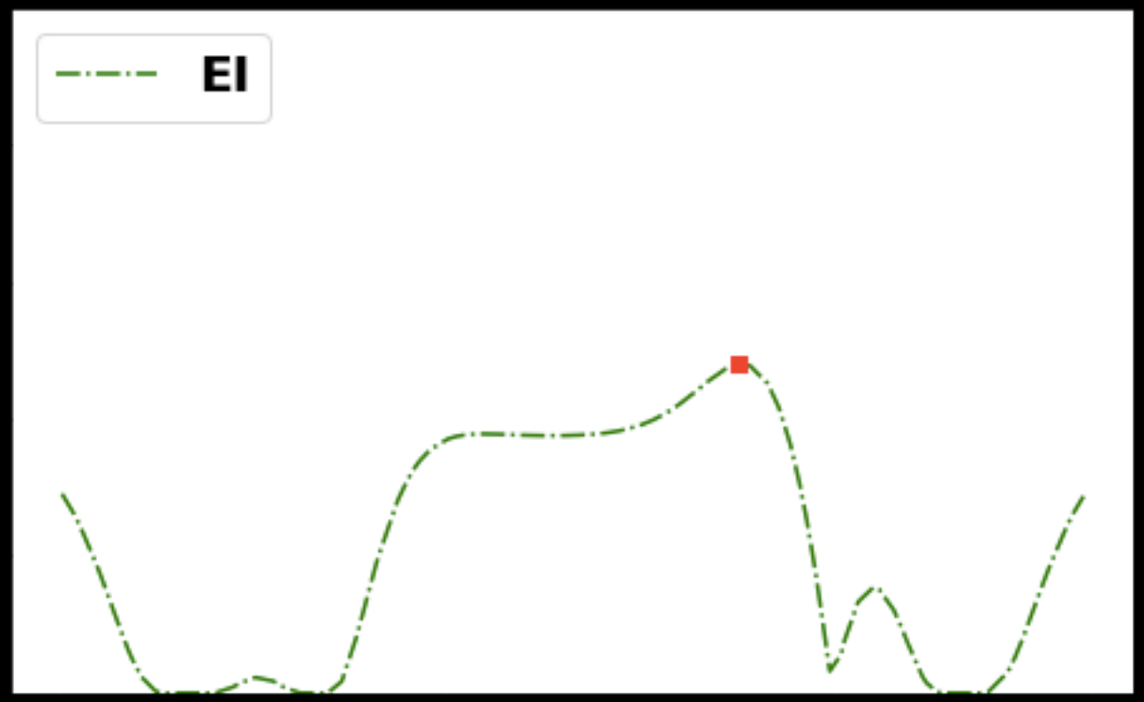




- We've evaluated $x^{(1)},\ldots x^{(n)}$, & observed $f(x^{(1)}),\ldots,f(x^{(n)})$ without noise.
- The best value observed is $f^*=\min(f(x^{(1)}),\ldots,f(x^{(n)}))$.
- If we evaluate at x, we observe f(x).
- The *improvement* is $(f^*-f(x))^+$
- The *expected improvement* is $EI(x)=E_n[(f^*-f(x))^+]$

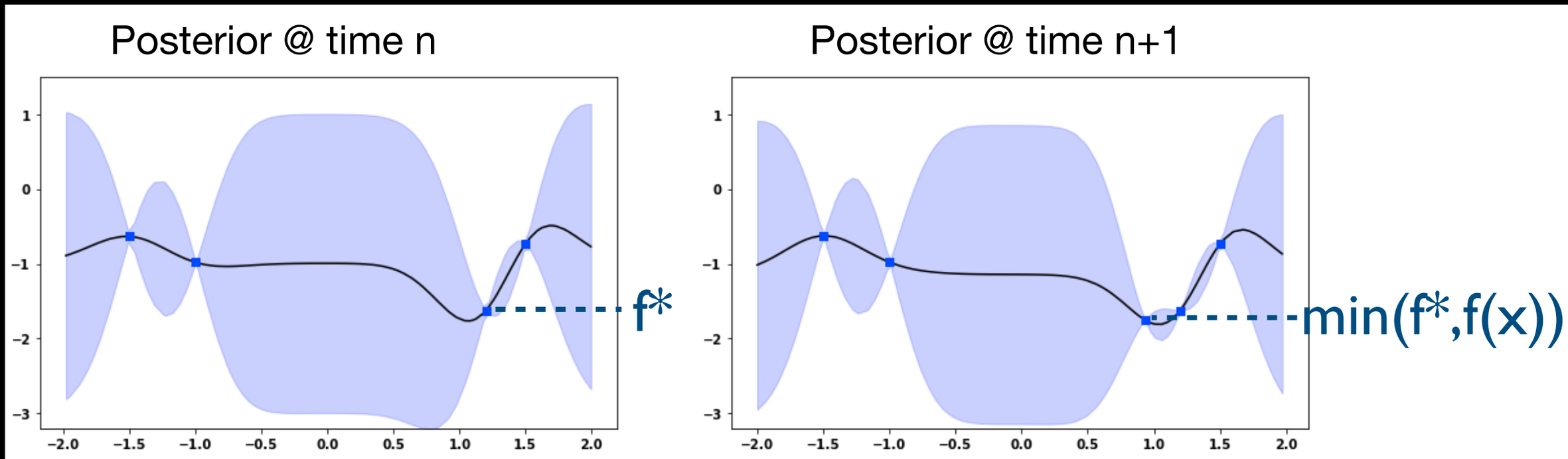
EI is one-step Bayes-optimal under assumptions





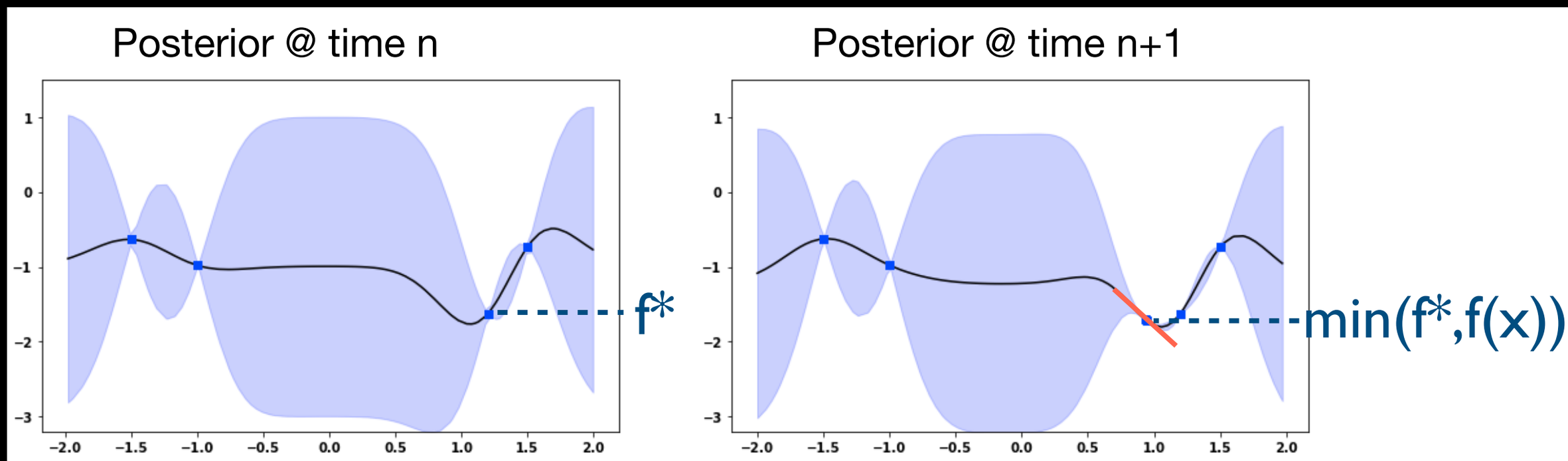
- One-step: We may evaluate f only one more time. After, we must report a solution.
- We are risk-neutral, and suffer loss equal to the value of the reported solution
- Evaluations are noise-free
- The solution we report must have known value

EI is one-step Bayes-optimal under assumptions

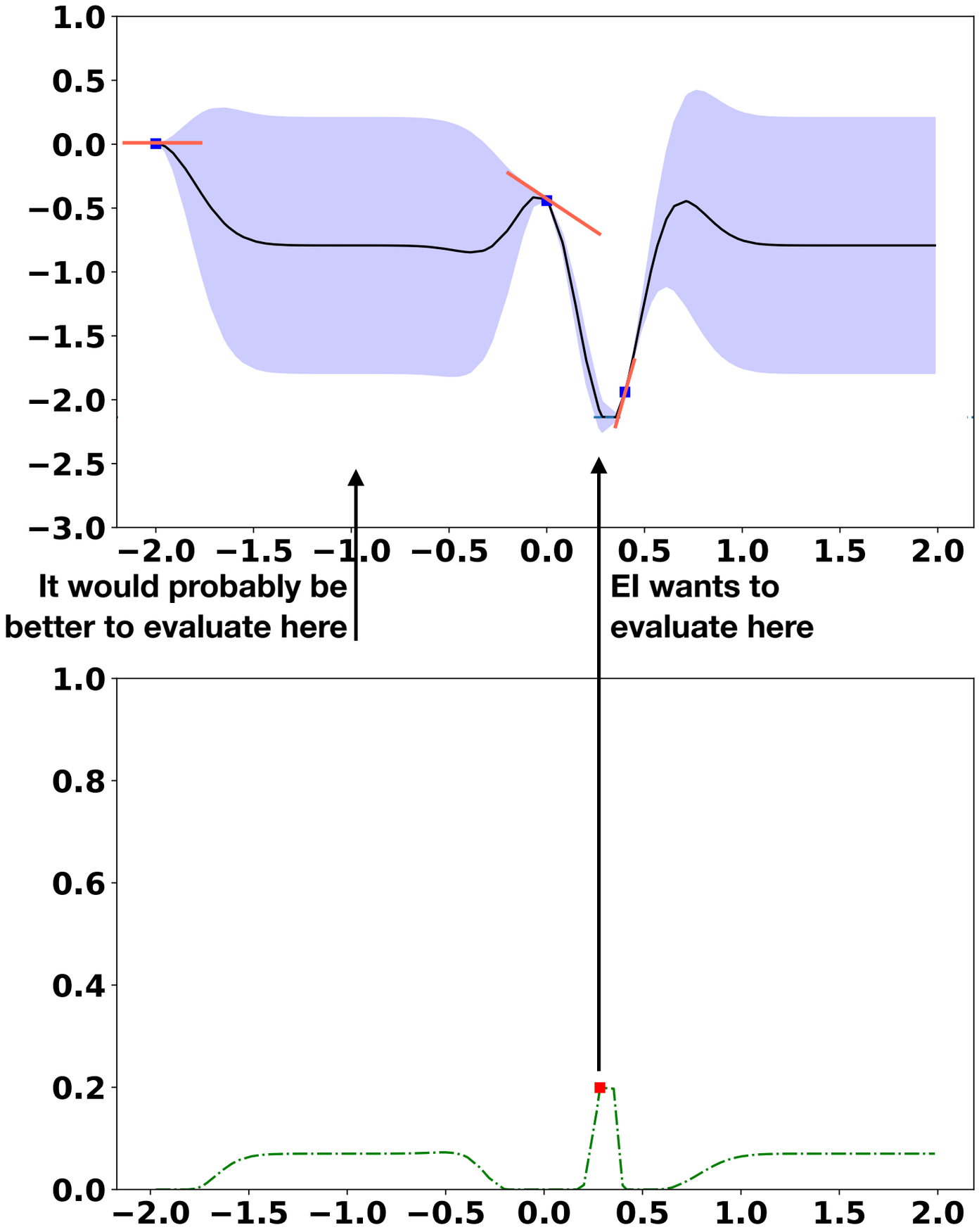


- Loss if we stop now:
  f*
- Loss if we stop after sampling f(x):
  min(f*,f(x))
- Reduction in loss due to sampling:
  $E_n[f^* - \min(f^*, f(x))] = E_n[(f^* - f(x))^+] = EI(x)$

EI places no value on some kinds of information

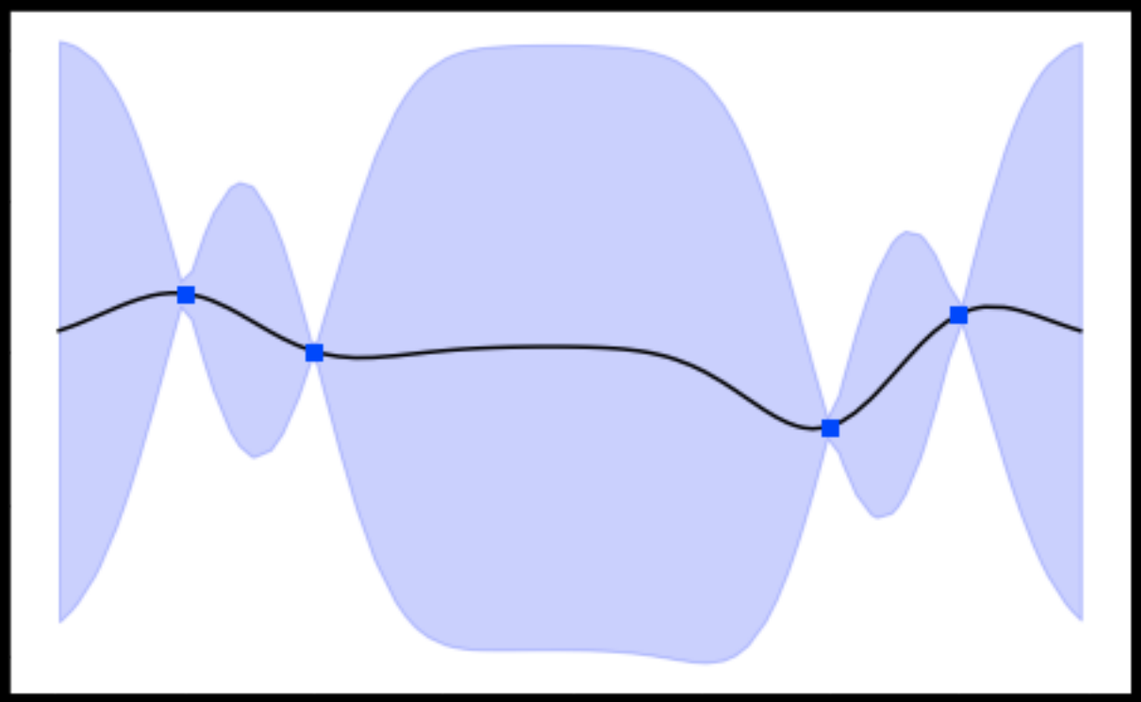


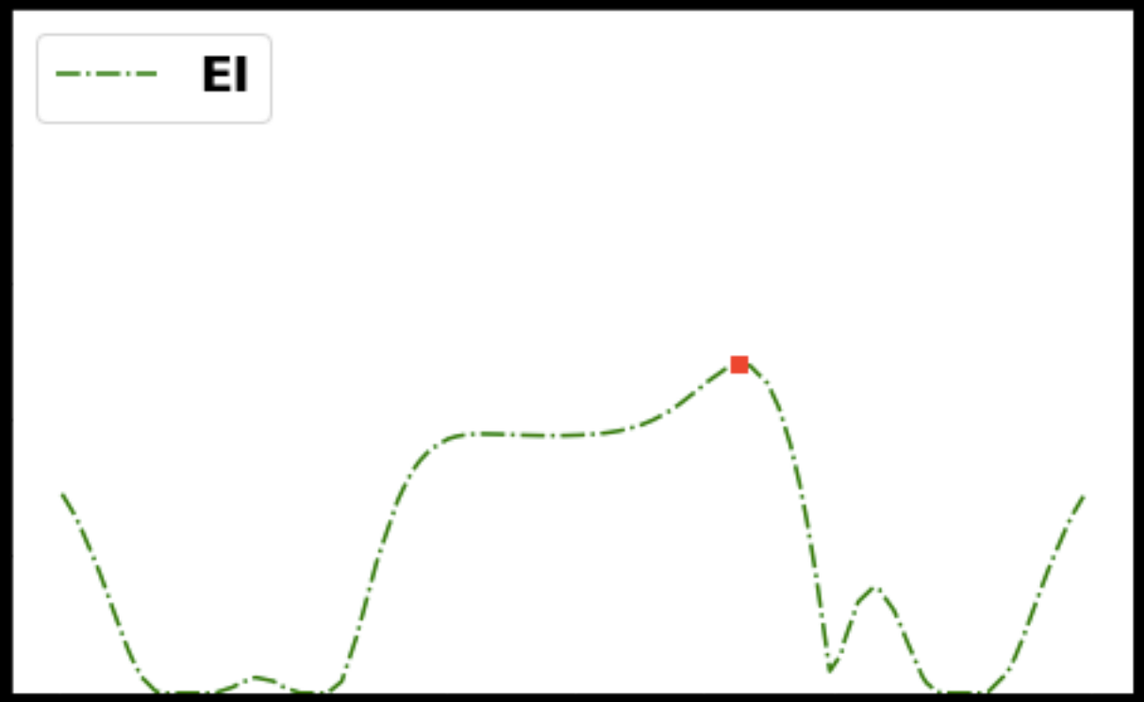
- Loss if we stop now:
  f*
- Loss if we stop after sampling f(x) & its gradient:
  min(f*,f(x))
- Reduction in loss due to sampling:
  $E_n[f^* - \min(f^*, f(x))] = E_n[(f^* - f(x))^+] = EI(x)$



As a consequence, EI can make poor decisions

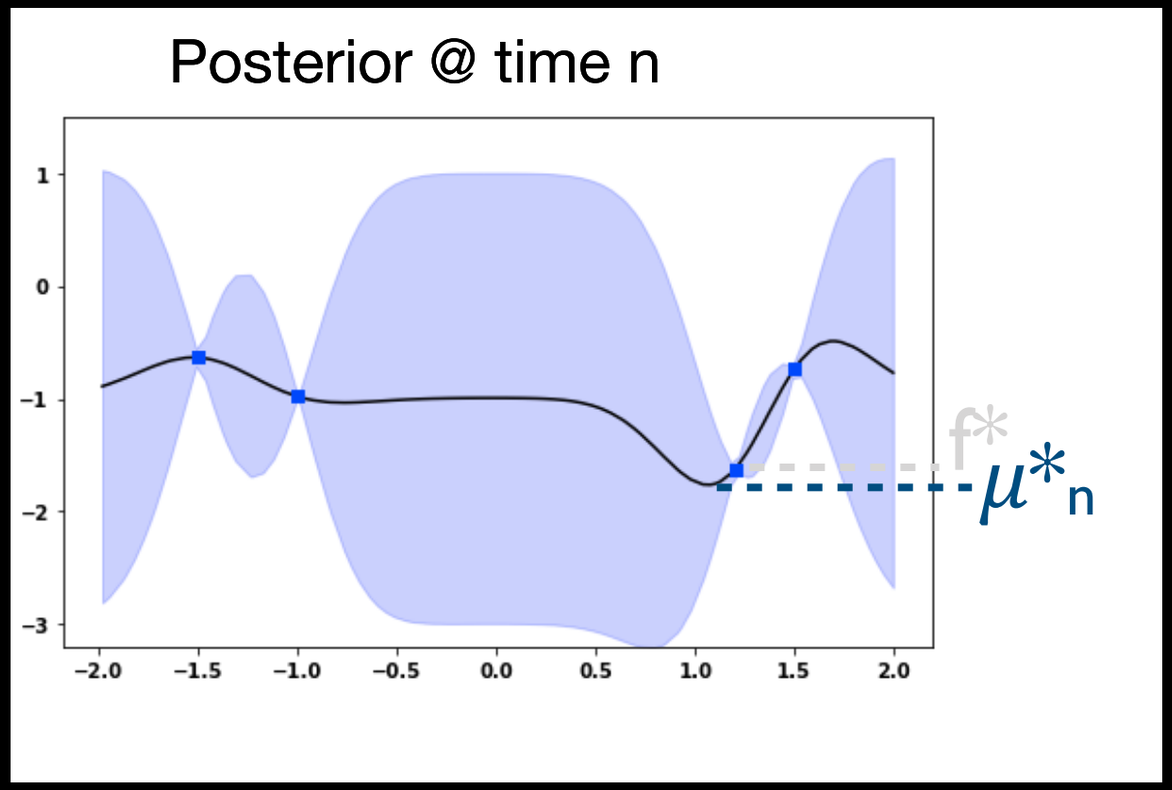
KG eliminates two of EI's assumptions





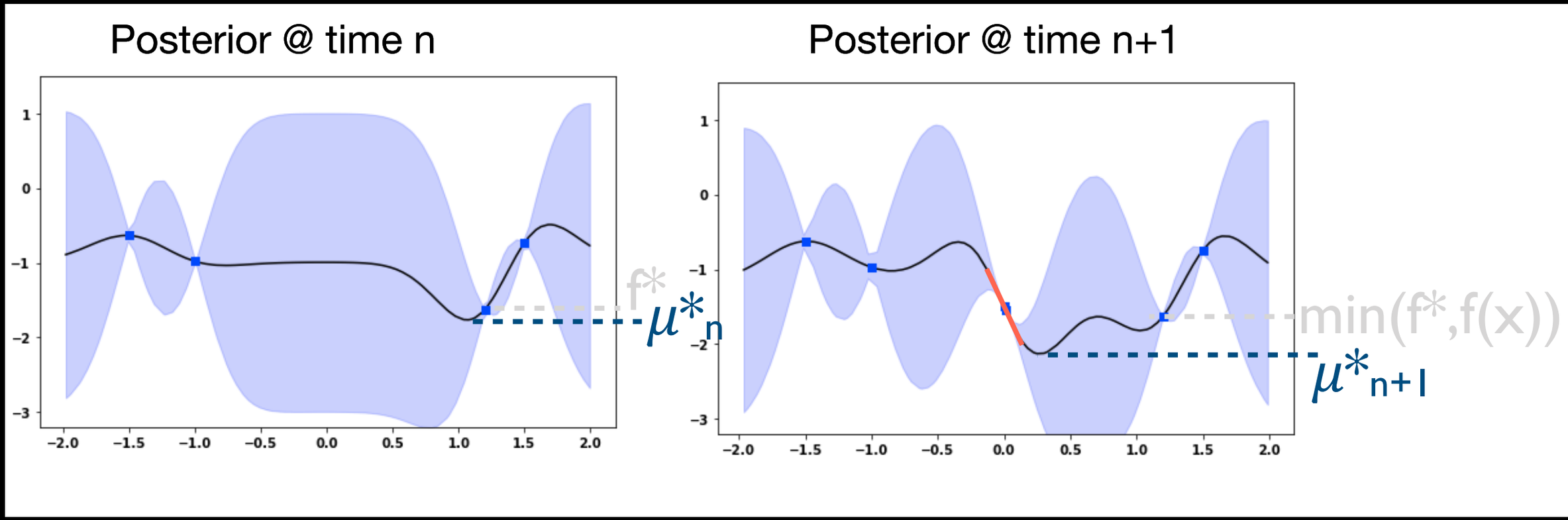
- One-step: We may evaluate f only one more time. After, we must report a solution.
- We are risk-neutral, and suffer loss equal to the value of the reported solution
- ~~Evaluations are noise-free~~
- ~~The solution we report must have known value~~

The KG acquisition function, KG(x) for an observation with gradients



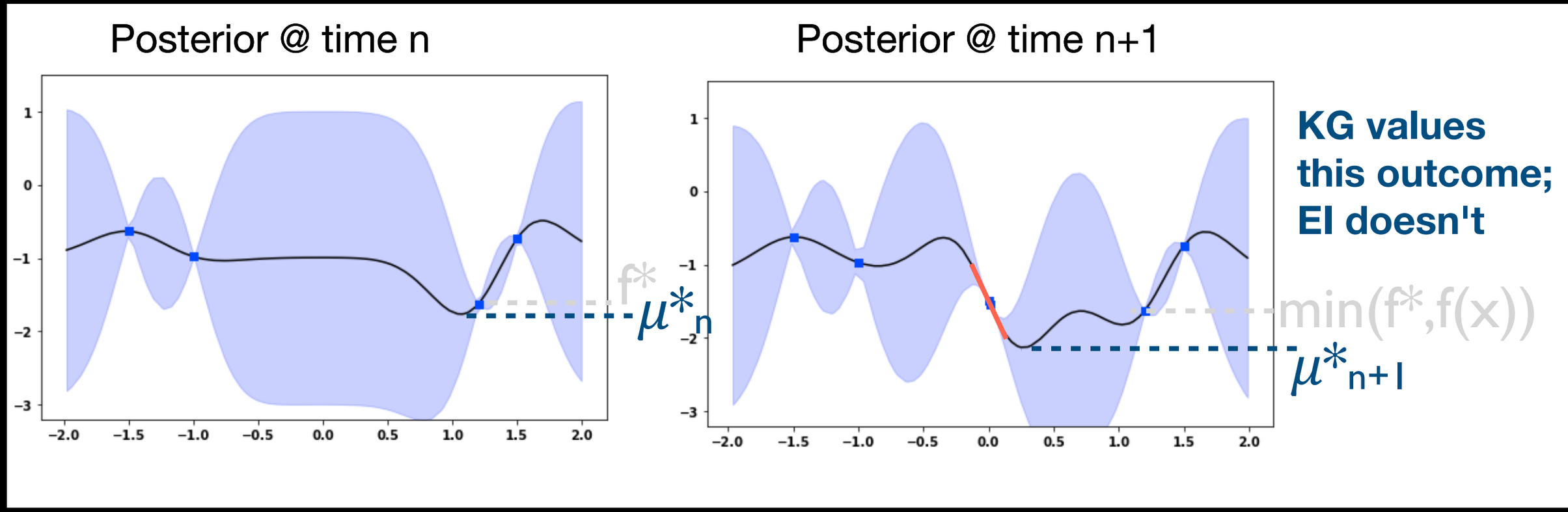
- Loss if we stop now:
  $\mu^*_n = \min_x \mu_n(x)$

The KG acquisition function, KG(x) for an observation with gradients



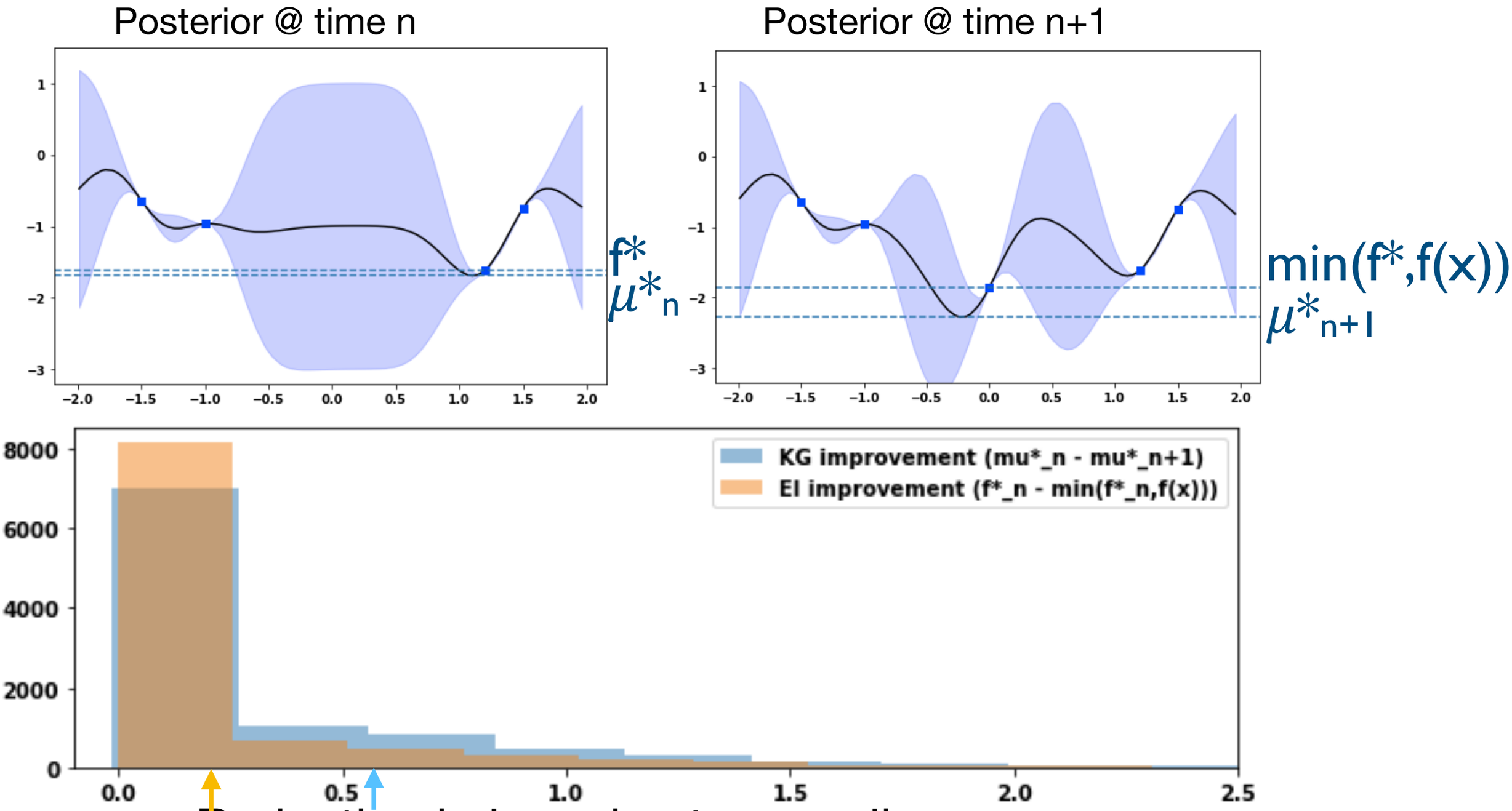
- Loss if we stop now:
  $\mu^*_n = \min_x \mu_n(x)$
- Loss if we stop after sampling f(x):
  $\mu^*_{n+1} = \min_x \mu_{n+1}(x)$

The KG acquisition function, KG(x) for an observation with gradients



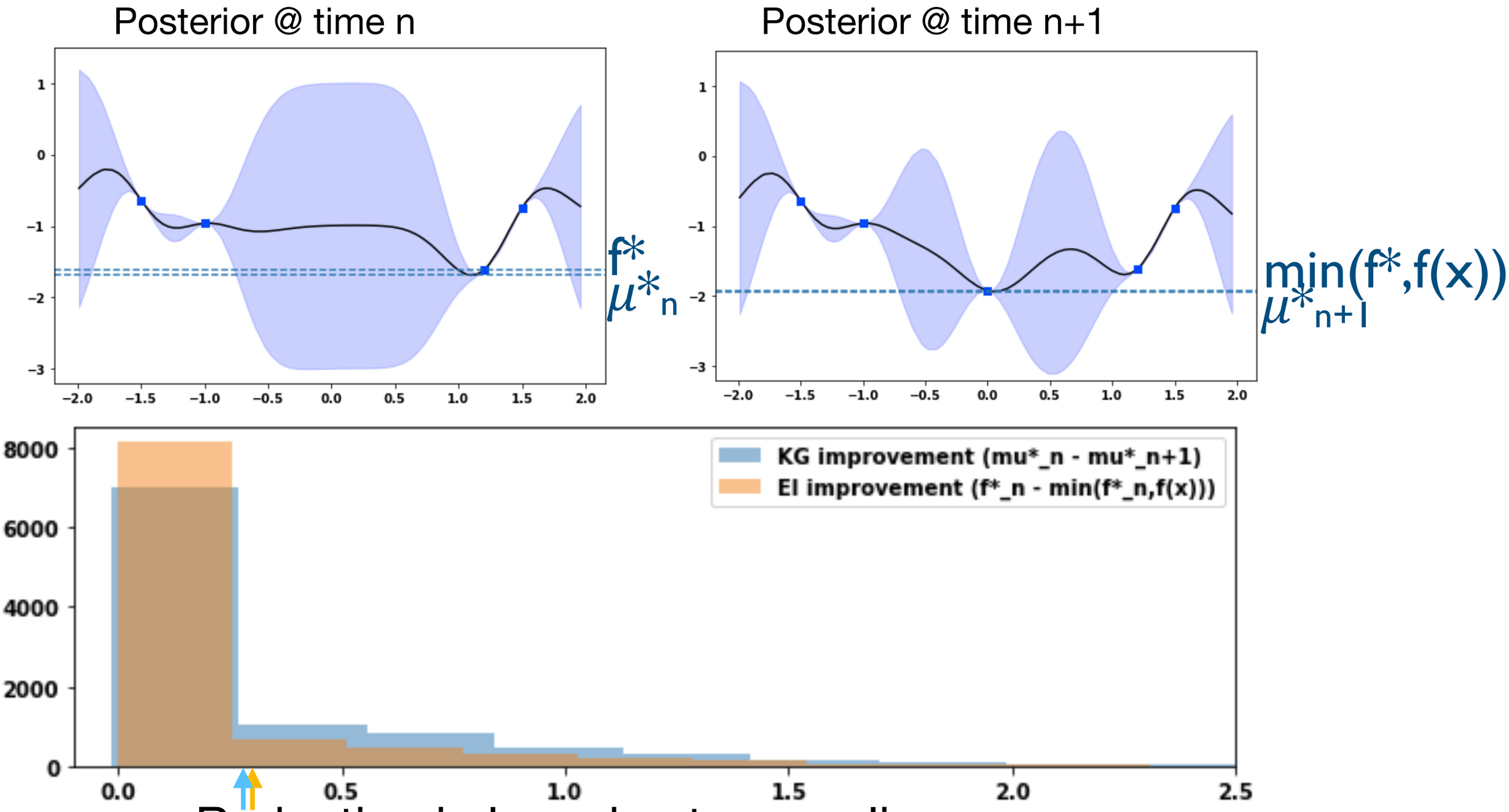
- Loss if we stop now:
  $\mu^*_n = \min_x \mu_n(x)$
- Loss if we stop after sampling f(x):
  $\mu^*_{n+1} = \min_x \mu_{n+1}(x)$
- Reduction in loss due to sampling:
  $KG(x) = E_n[\mu^*_n - \mu^*_{n+1} \mid \text{query } x]$

Here's how we can compute the KG acquisition function for BO with gradients



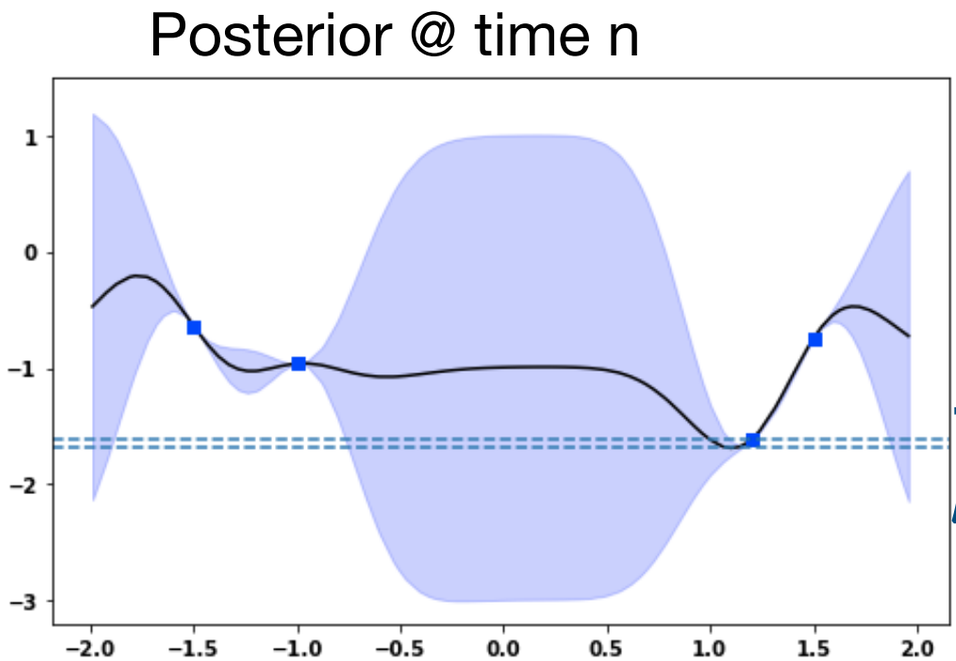
- Reduction in loss due to sampling: $E_n[\mu^*_n - \mu^*_{n+1}]$

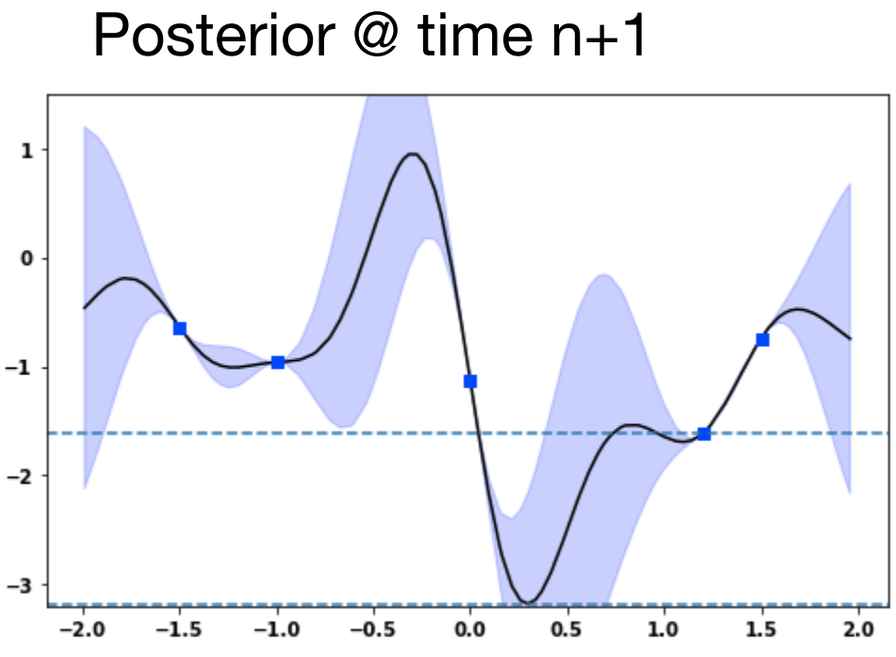
Here's how we can compute the KG acquisition function for BO with gradients



- Reduction in loss due to sampling: $E_n[\mu^*_n - \mu^*_{n+1}]$

Here's how we can compute the KG acquisition function for BO with gradients

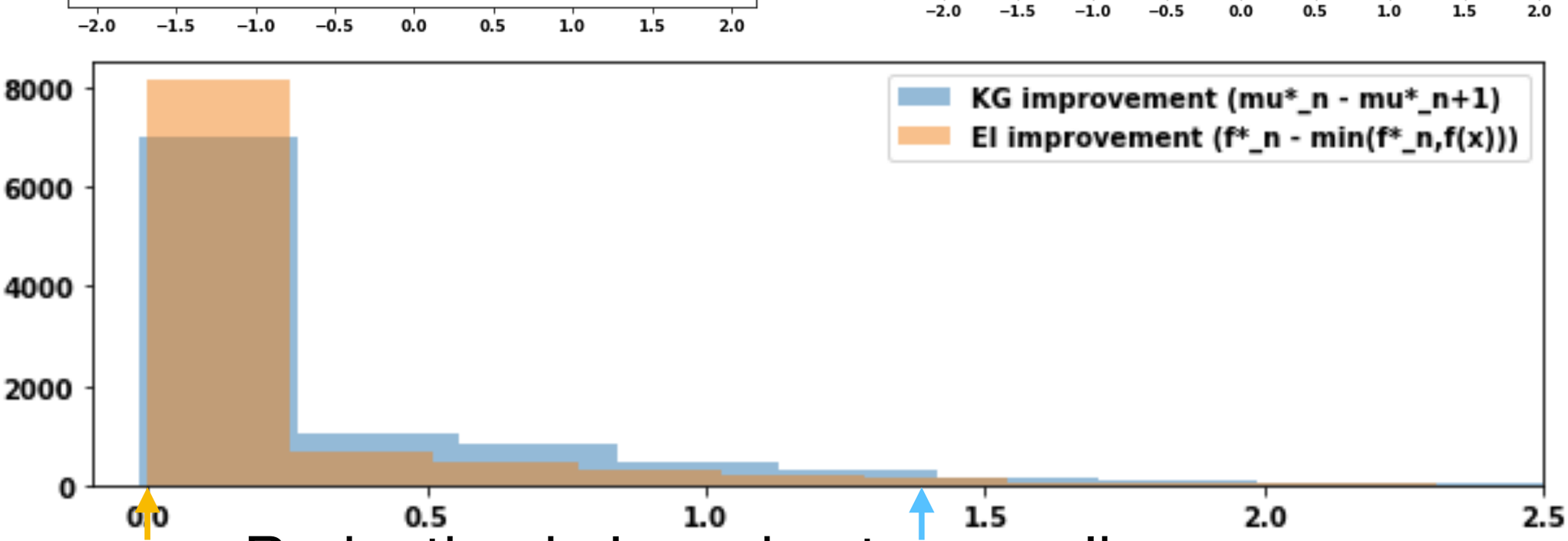




f*
μ^*_n

min(f*,f(x))

μ^*_{n+1}



- Reduction in loss due to sampling:
  $E_n[\mu^*_n - \mu^*_{n+1}]$

Here's basic pseudocode for computing the KG acquisition function

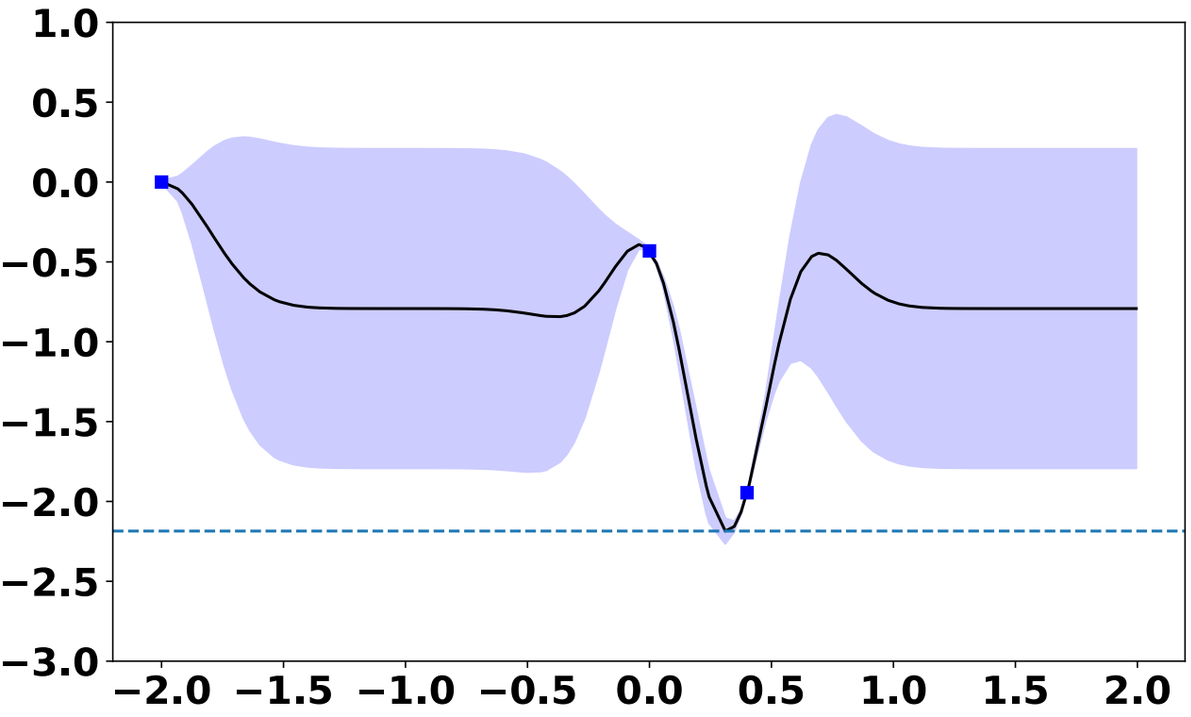
For i in 1:replications

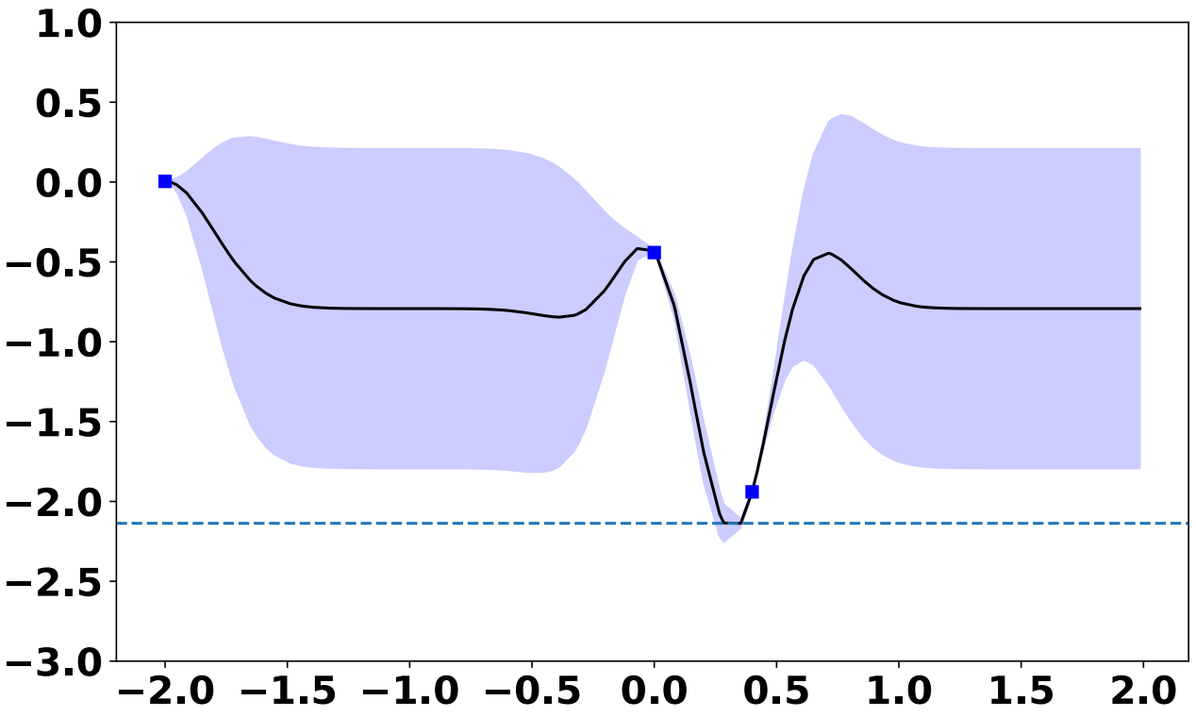
- Simulate f(x), ∇f(x) from the posterior
- Calculate $\mu^*_{n+1} = \min_{x'} \mu_{n+1}(x')$ from sim'd f(x), ∇f(x)
- Calculate $\mu^*_n - \mu^*_{n+1}$

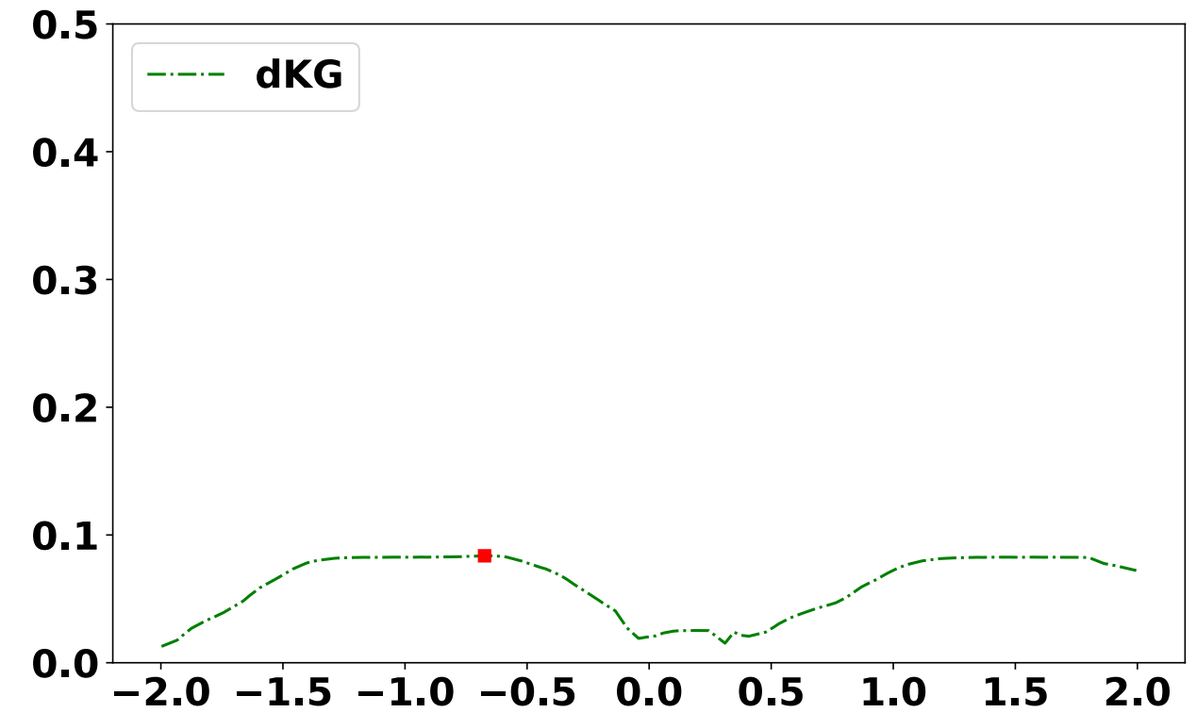
KG(x) is the average of the simulated $\mu^*_n - \mu^*_{n+1}$

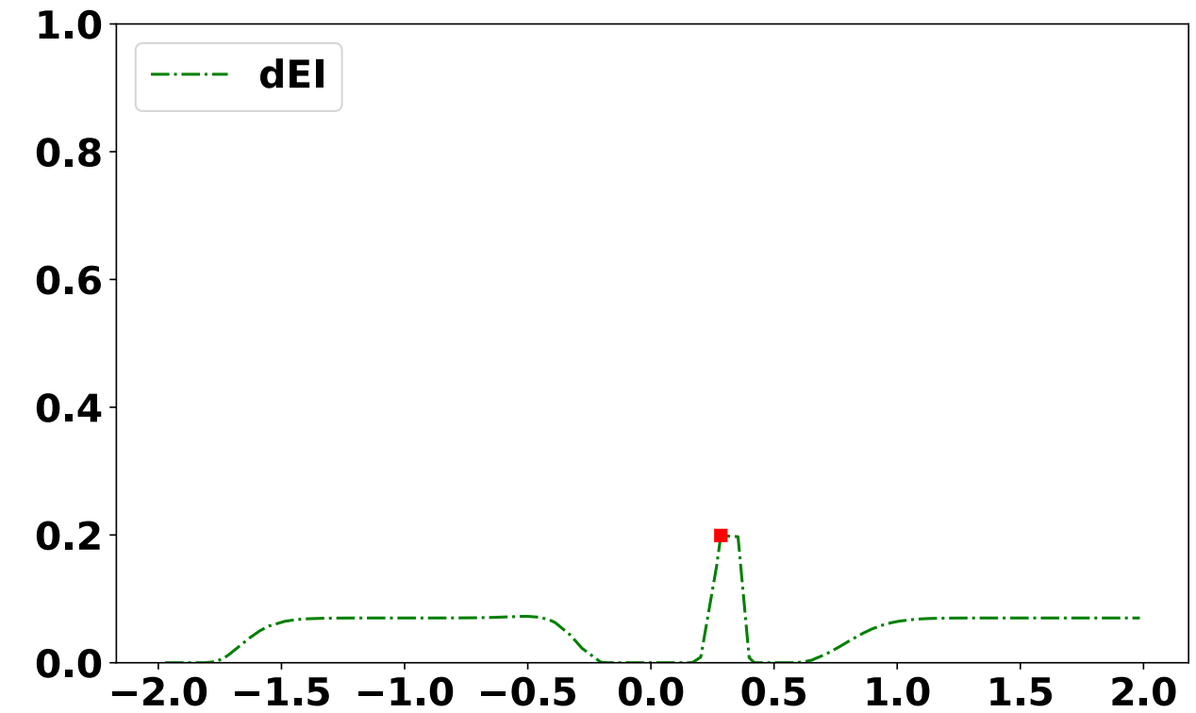
* I'll discuss a faster computational method in a few minutes

KG explores more effectively than EI when we have gradients

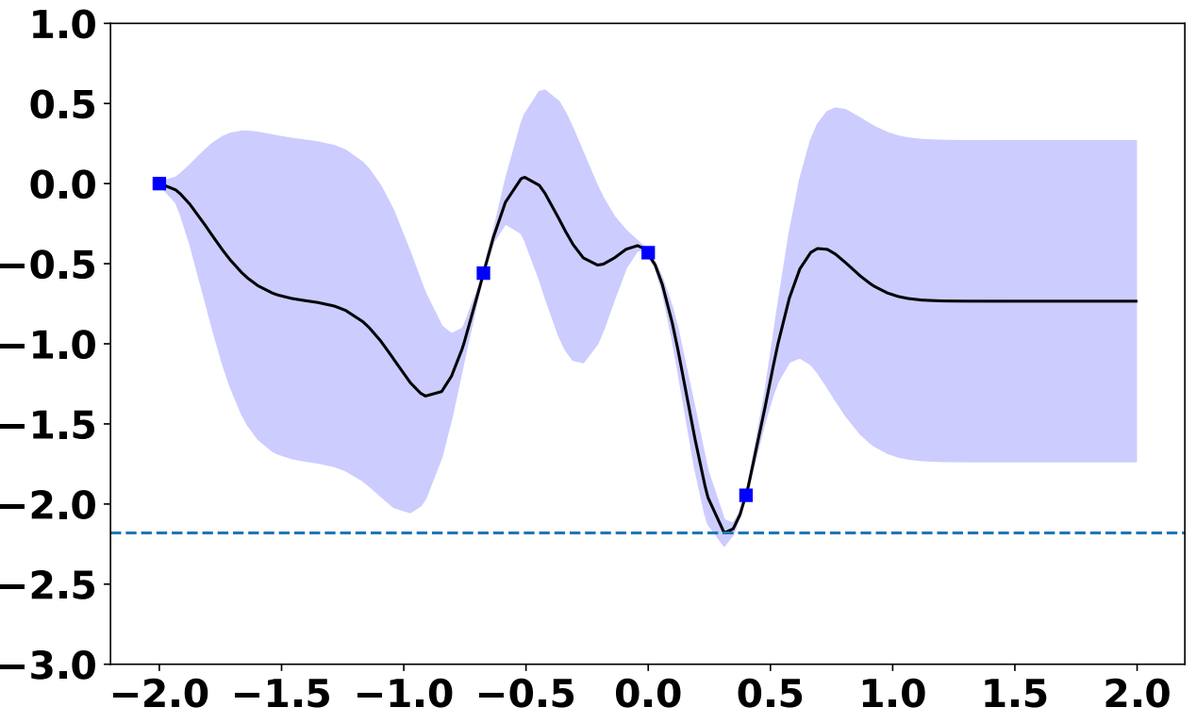


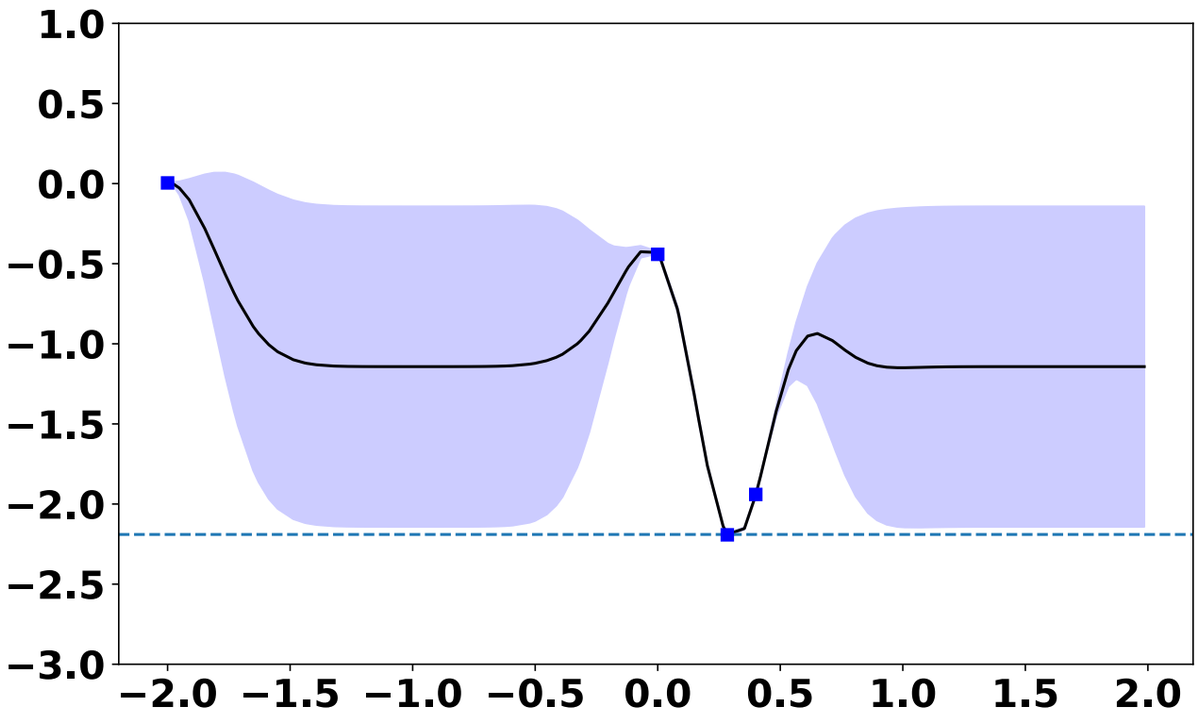


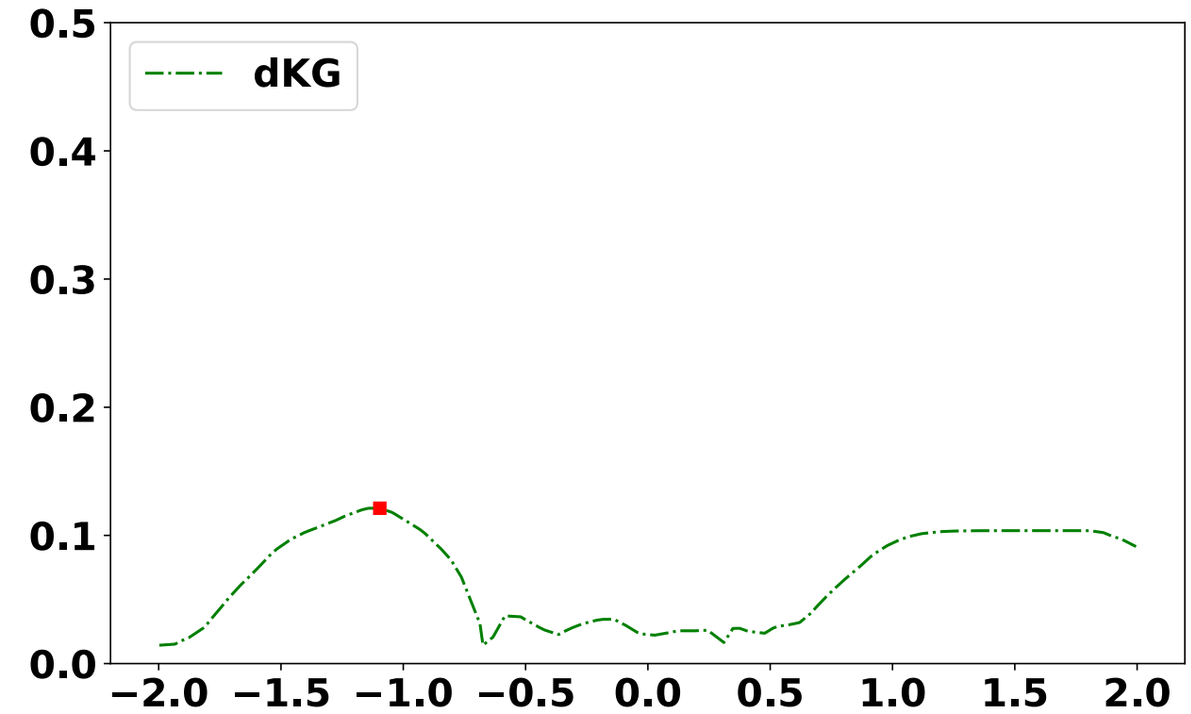


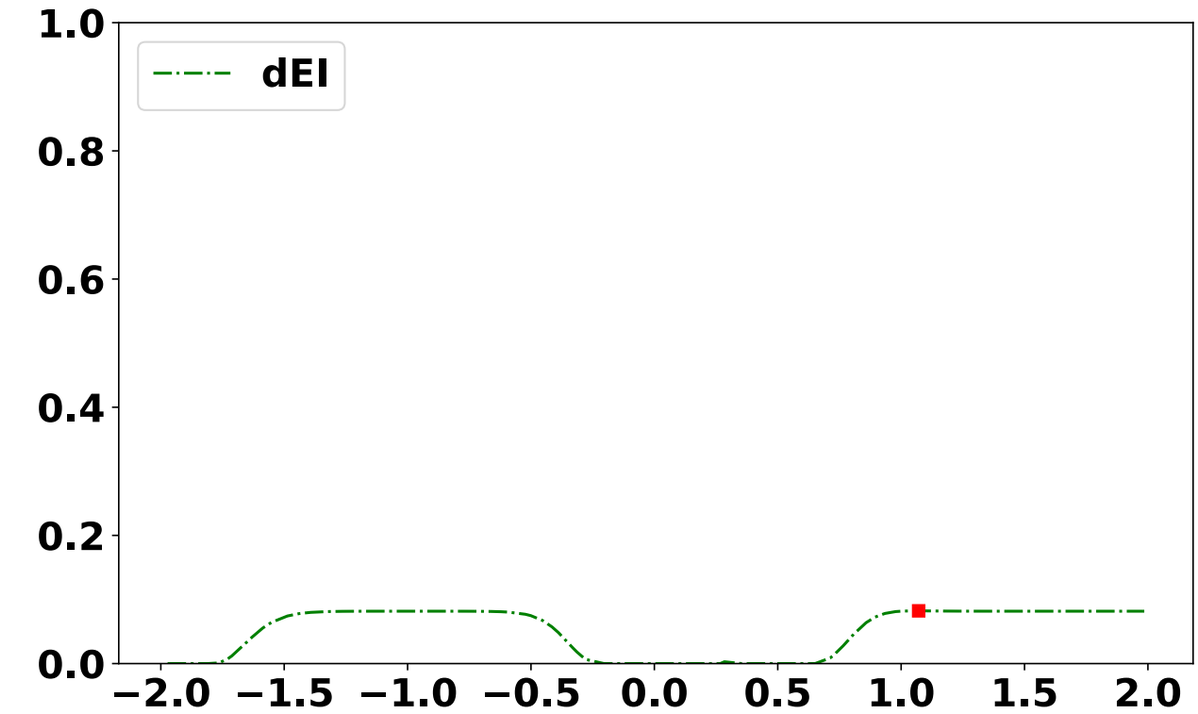


KG explores more effectively than EI when we have gradients

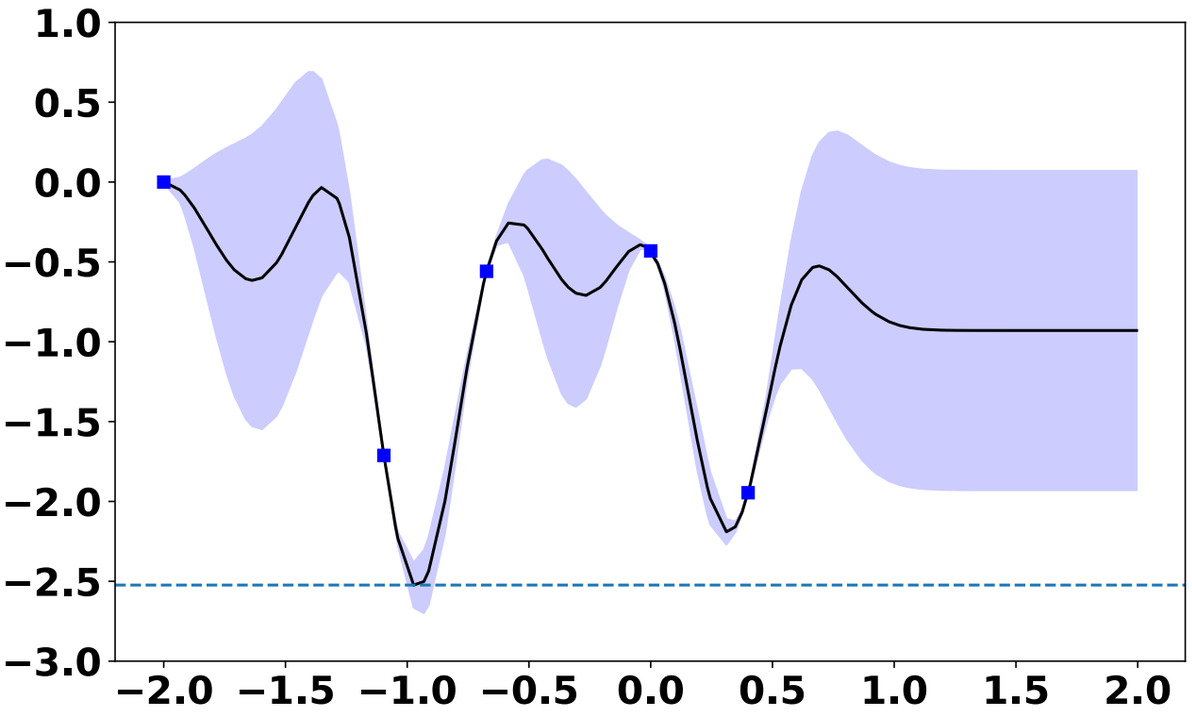


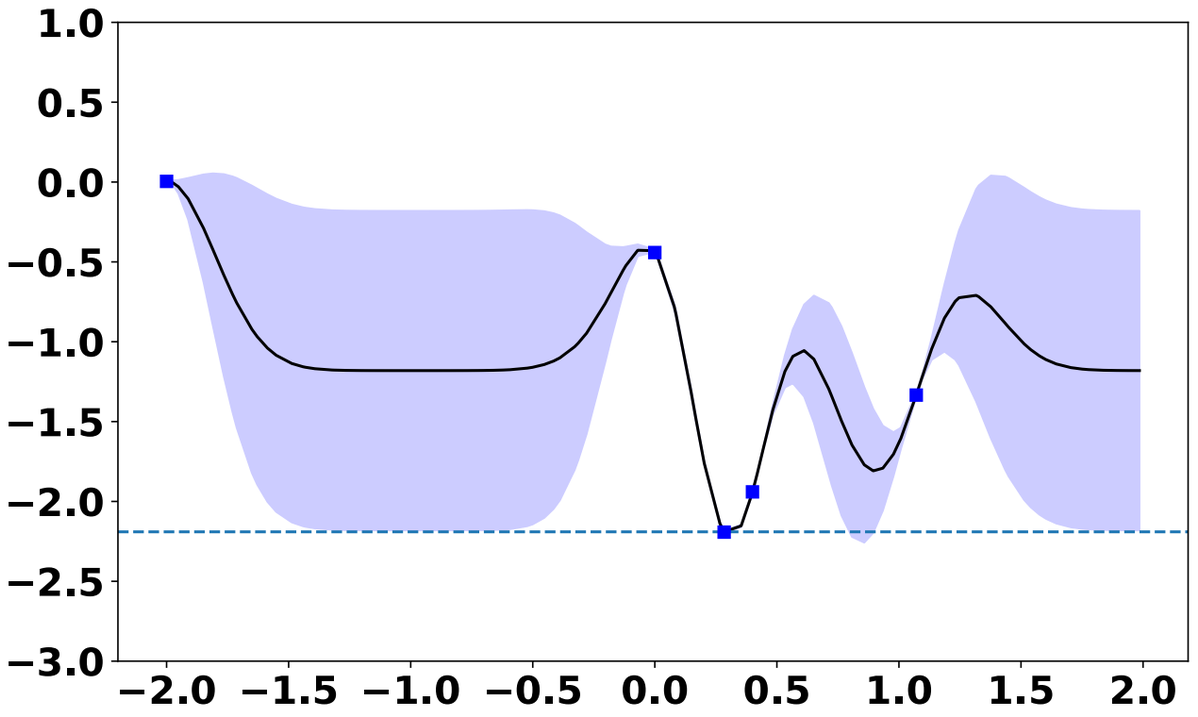


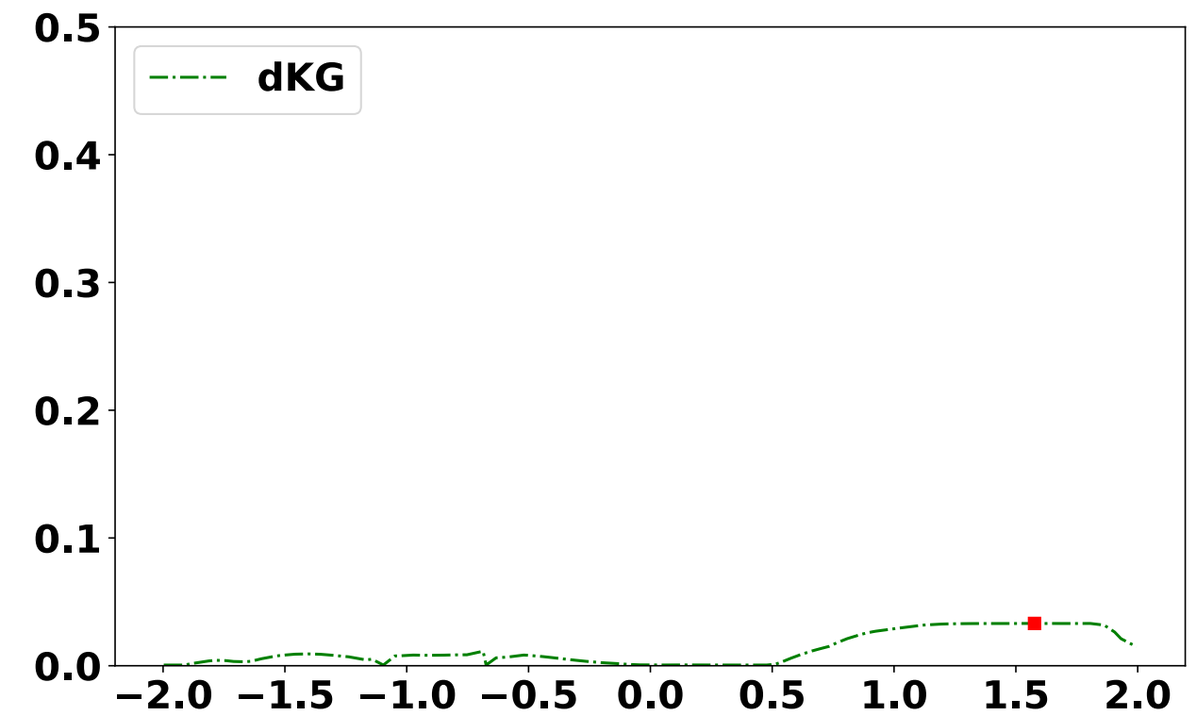


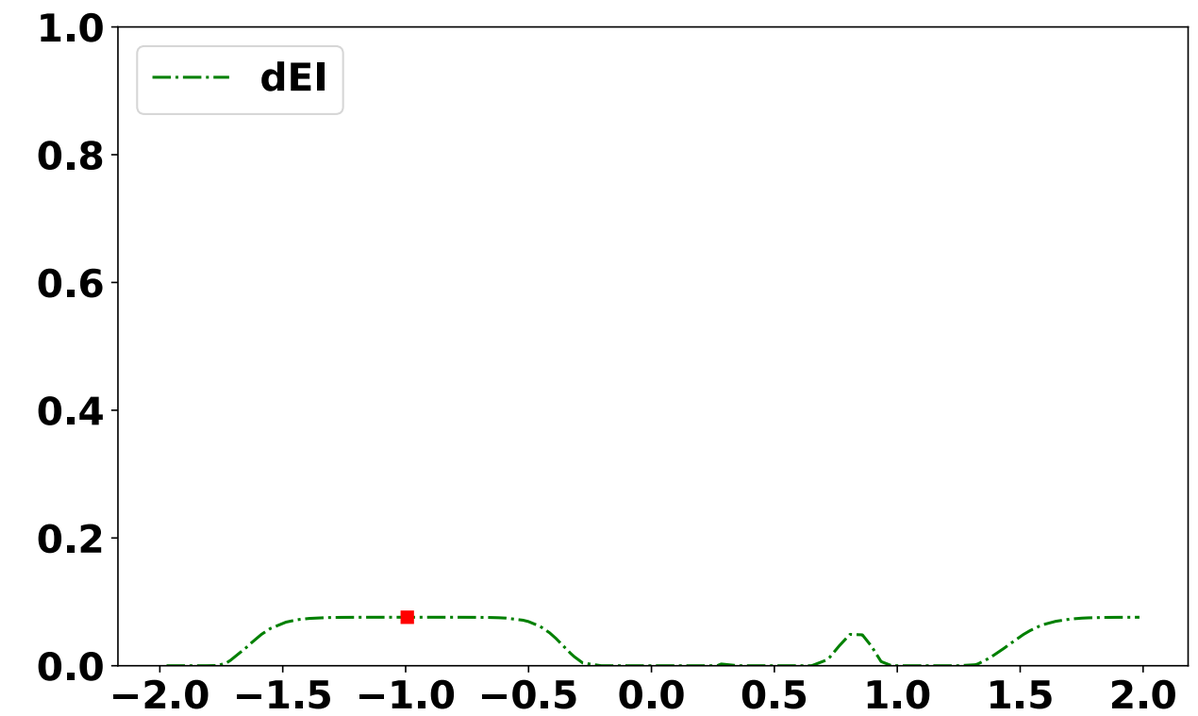


KG explores more effectively than EI when we have gradients

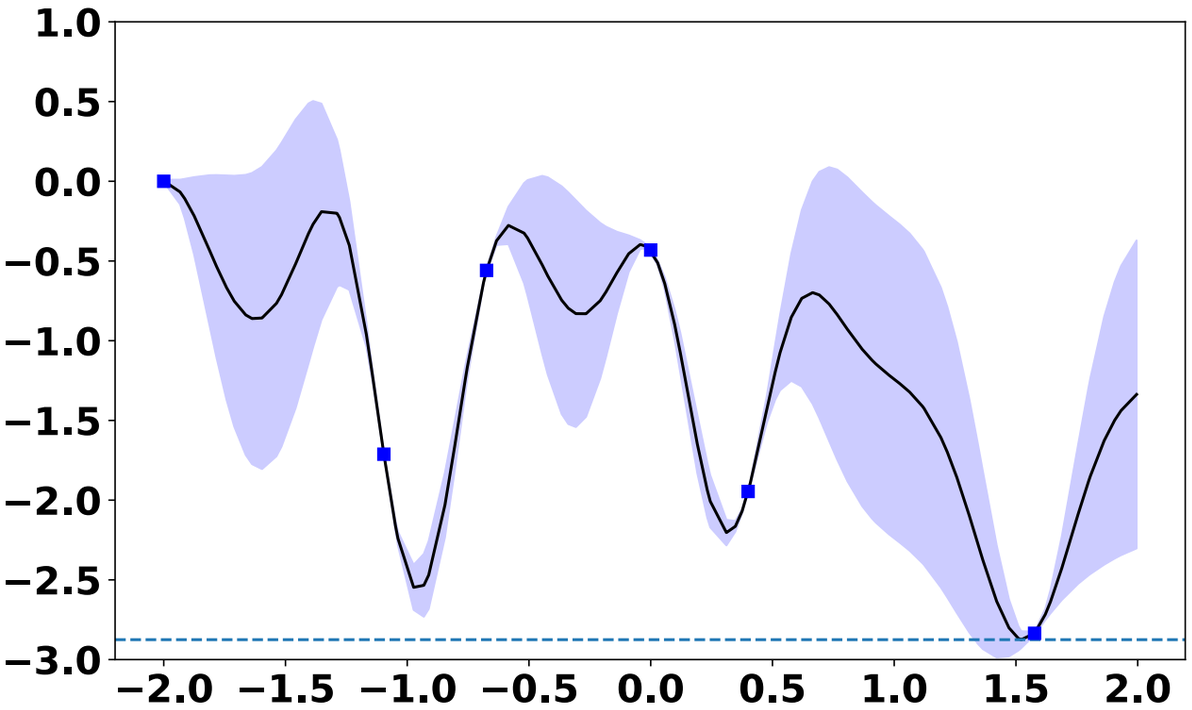


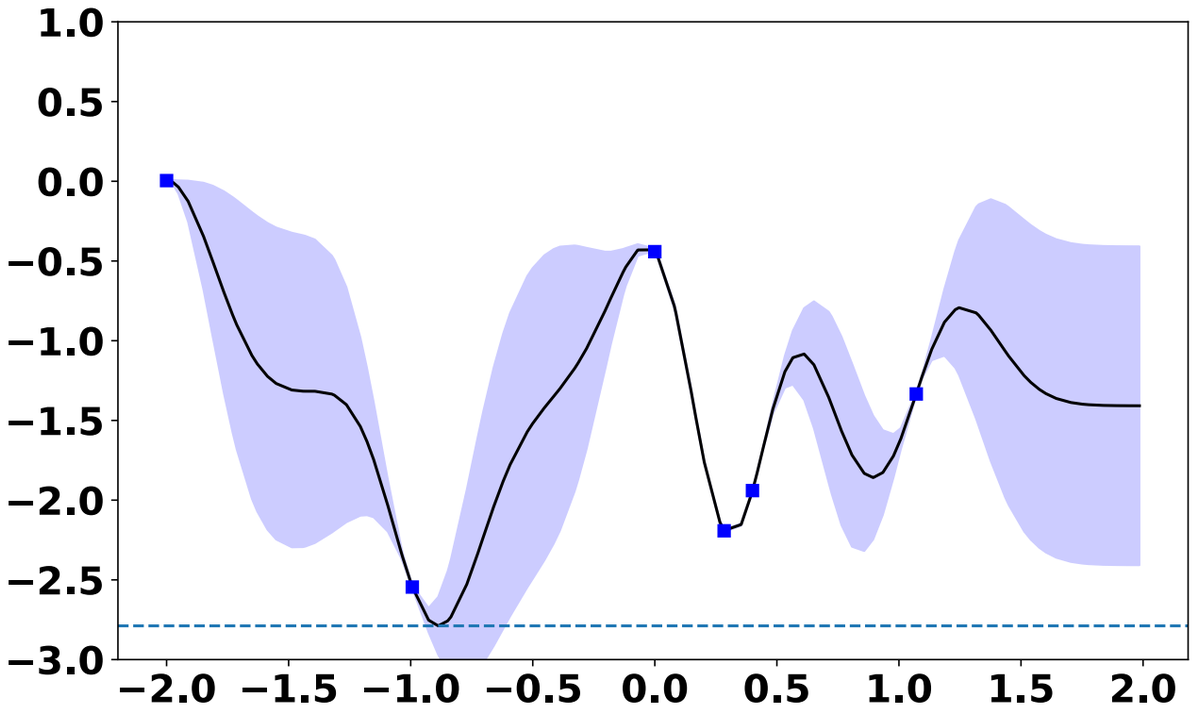


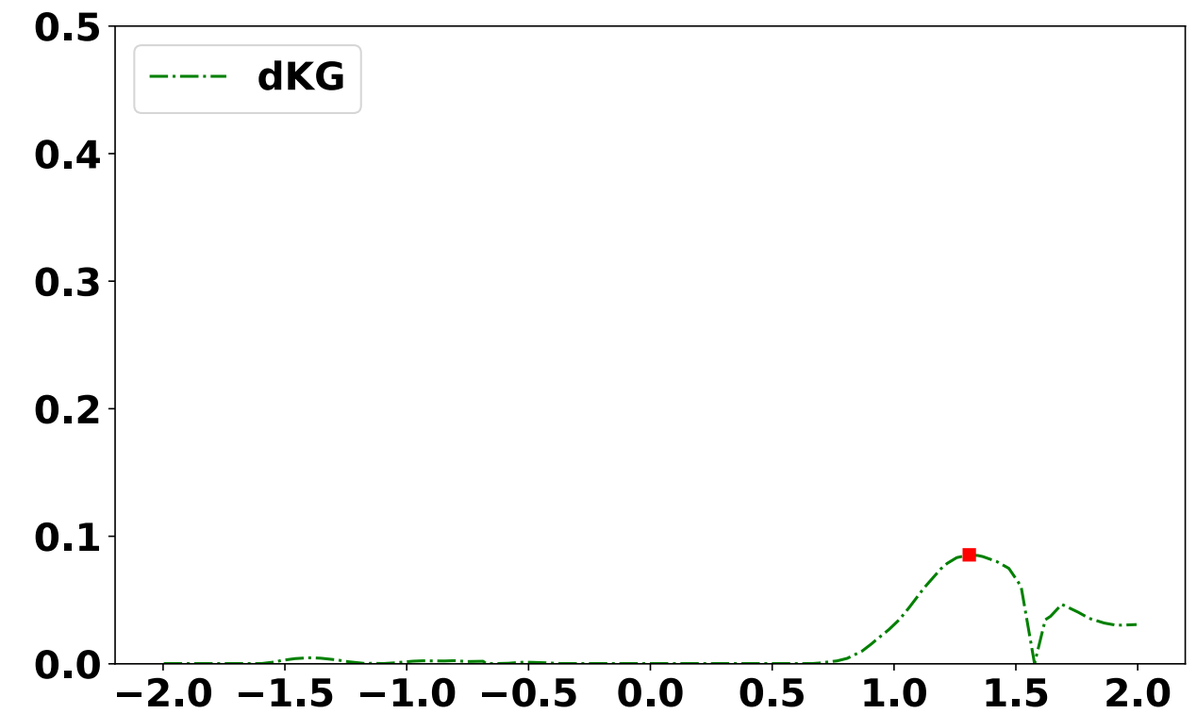


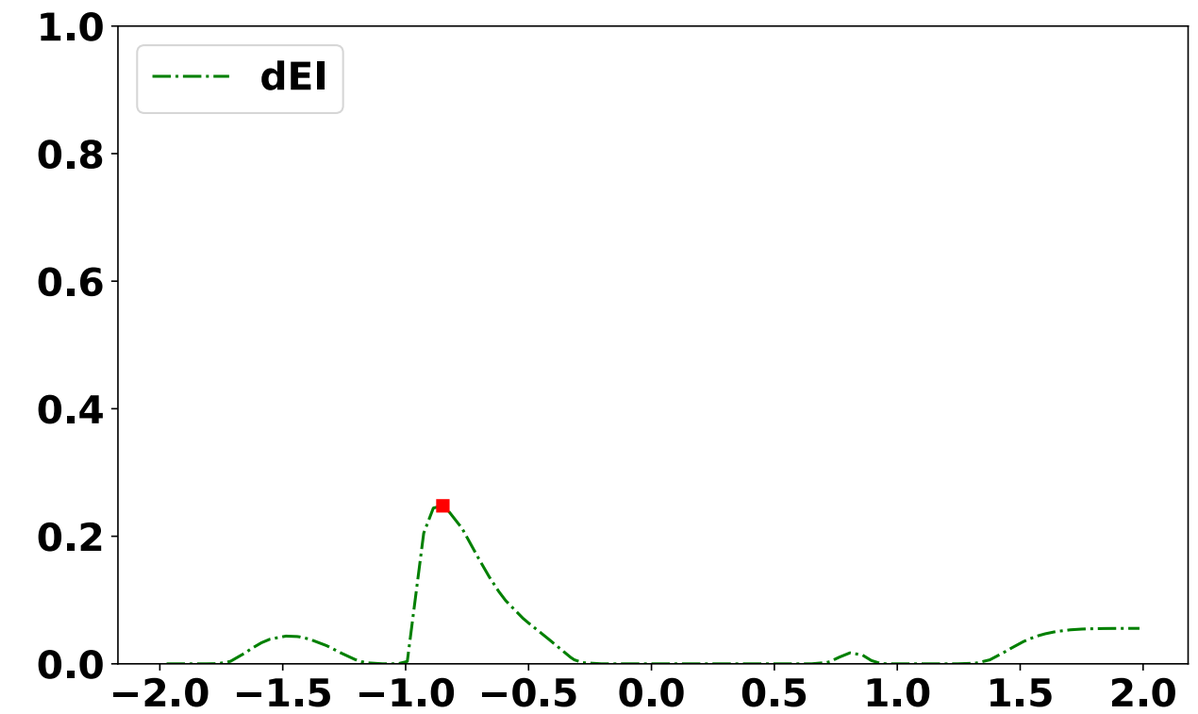


KG explores more effectively than EI when we have gradients

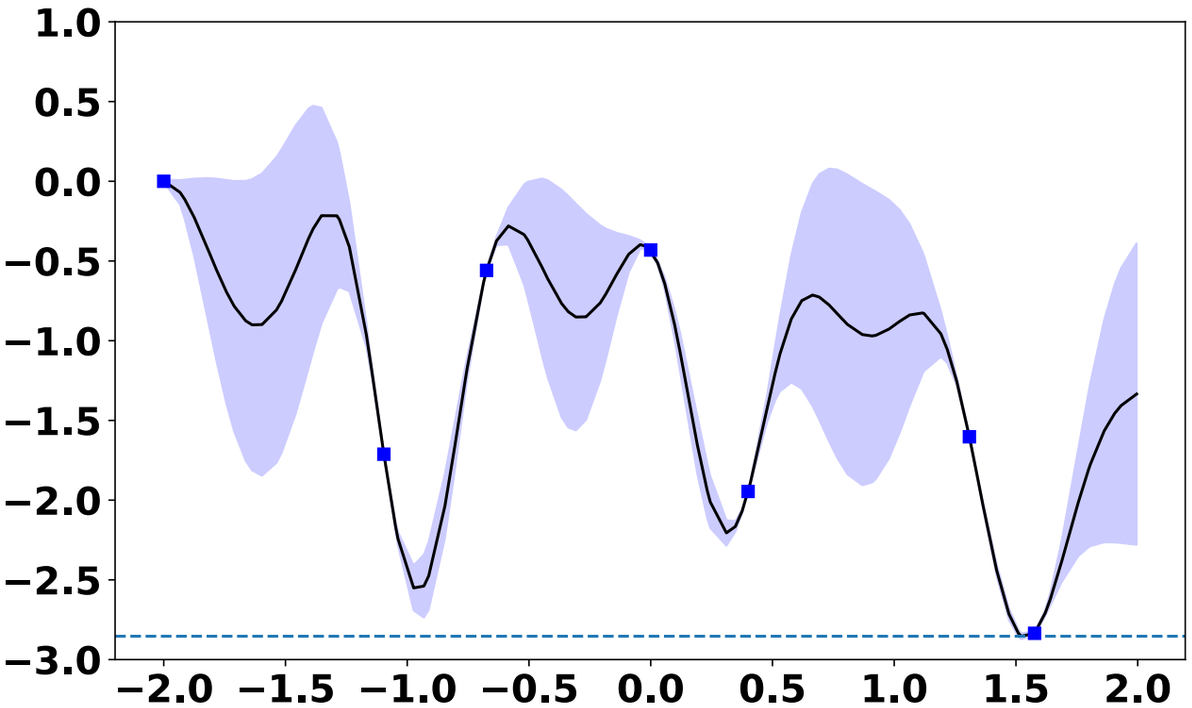


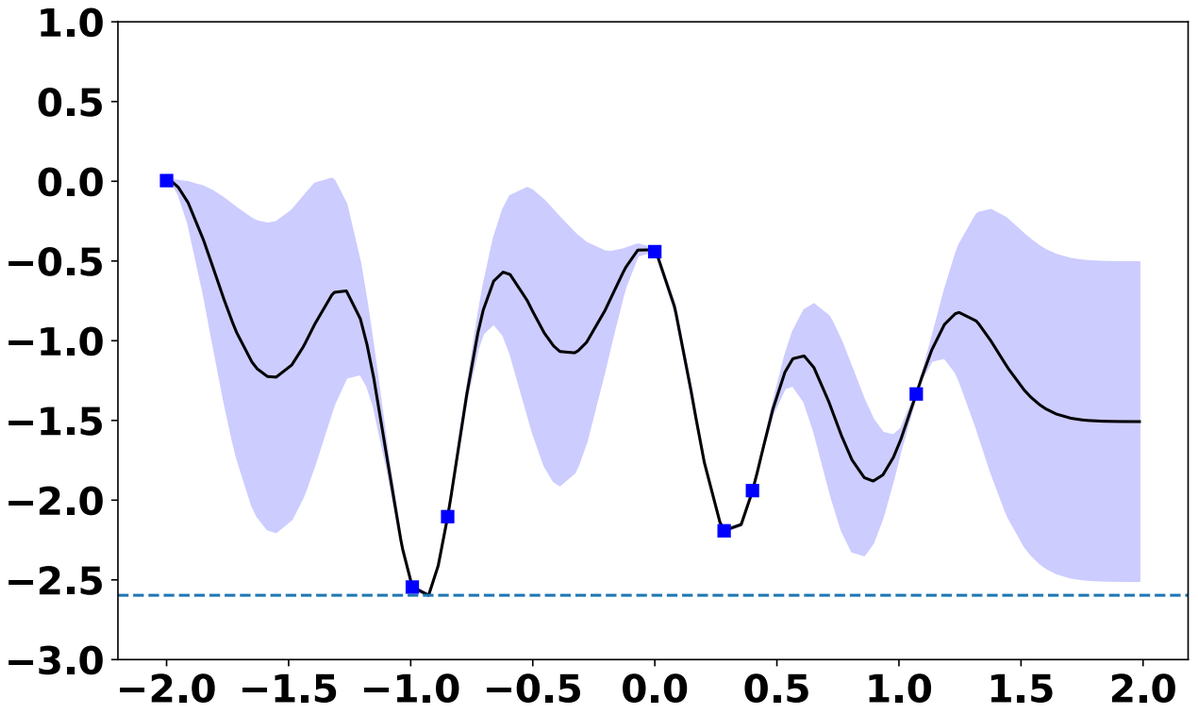


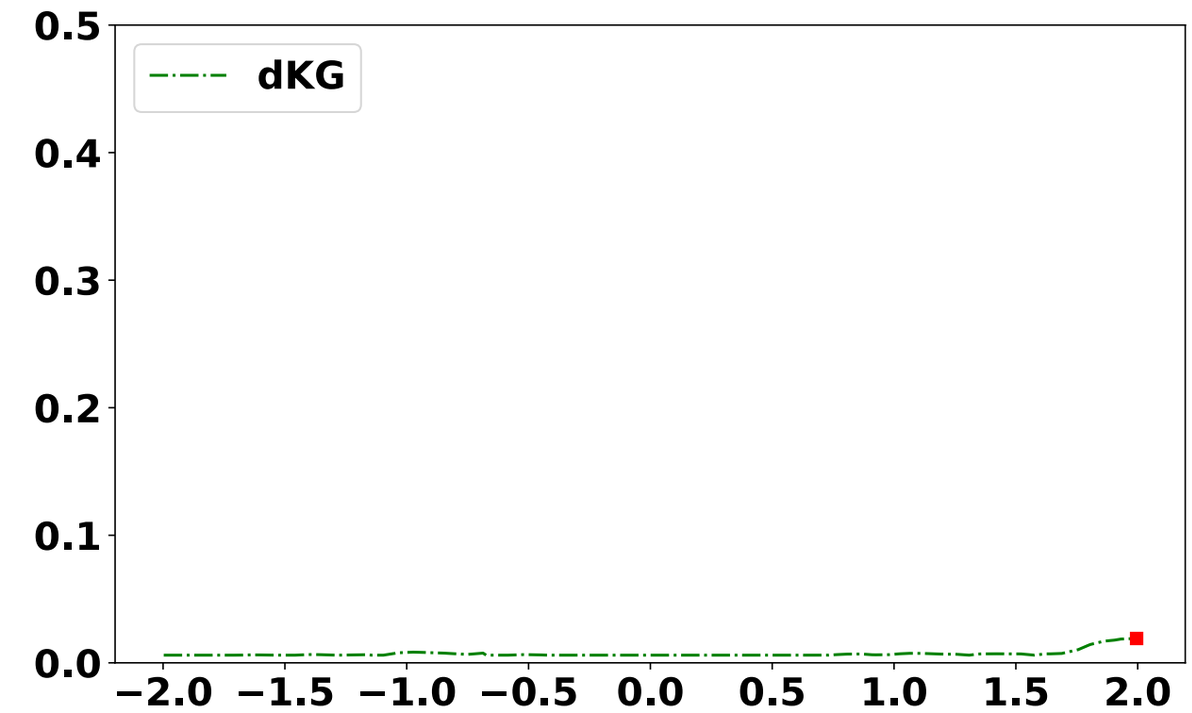


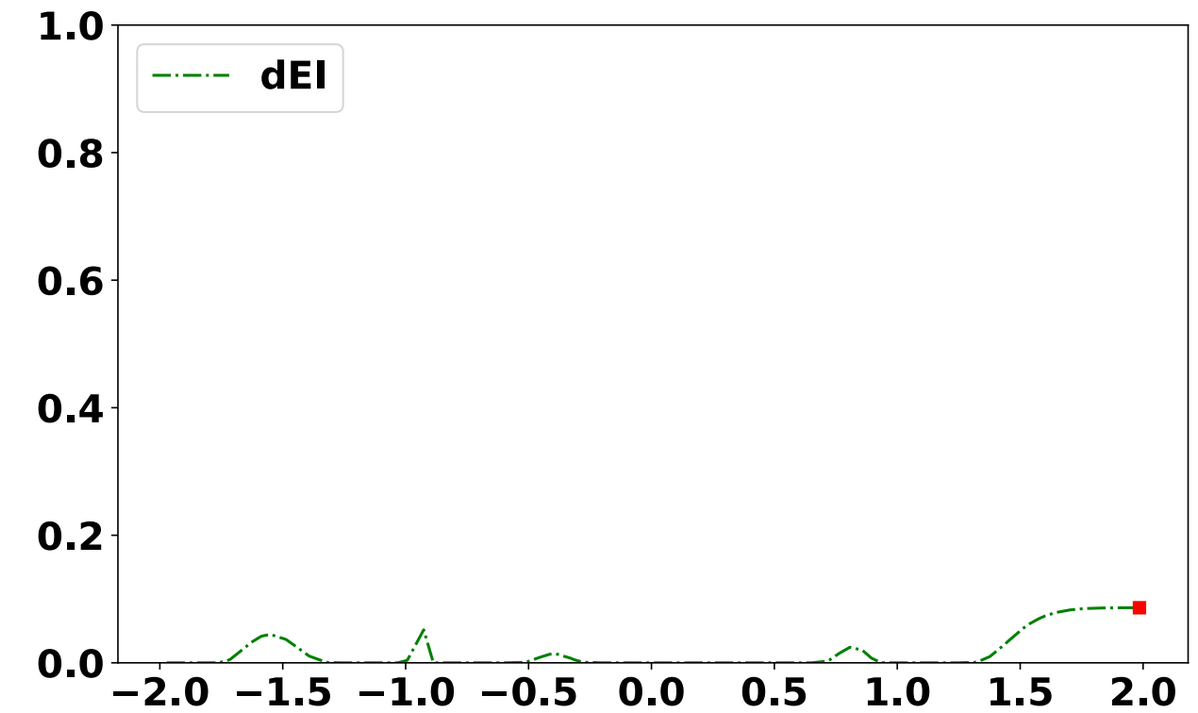


KG explores more effectively than EI when we have gradients

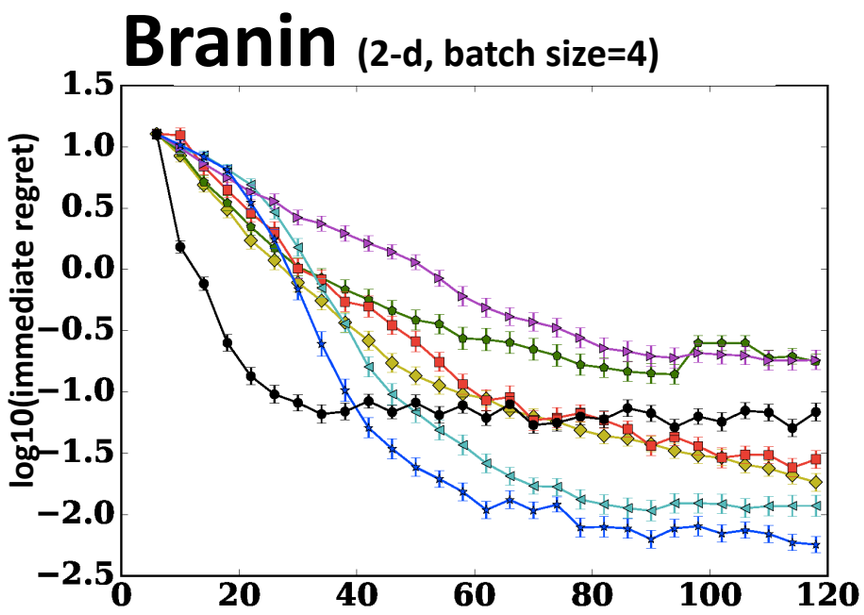


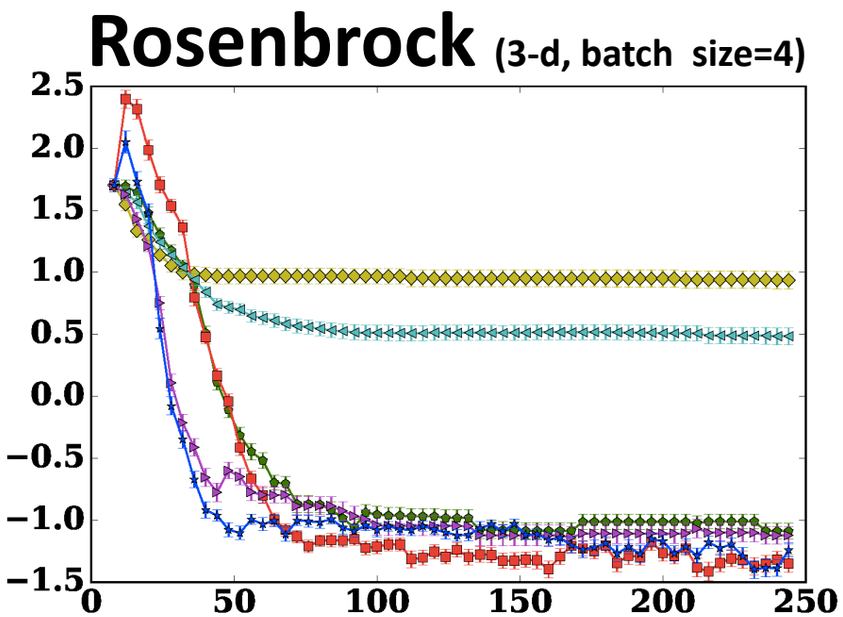


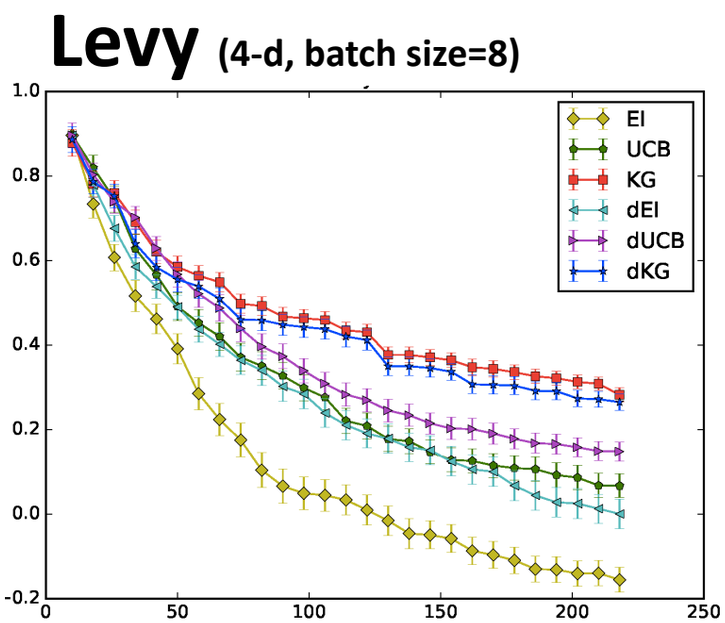


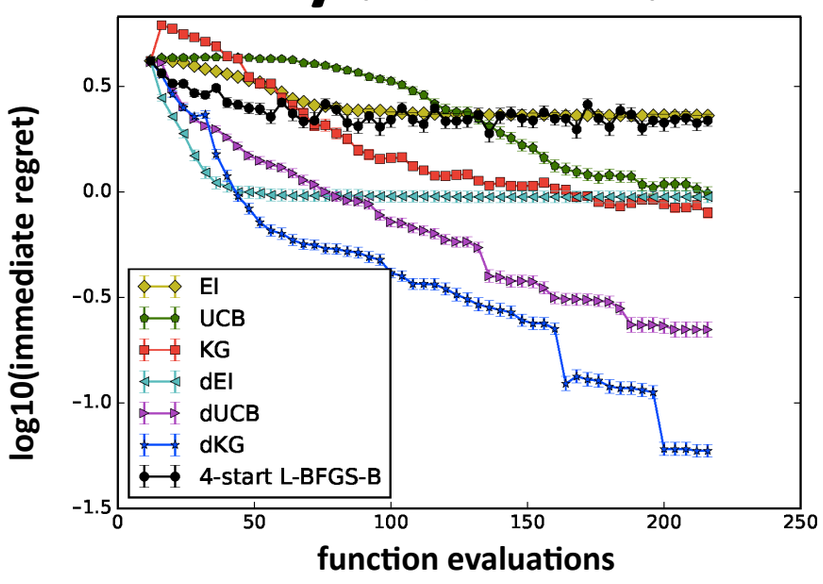


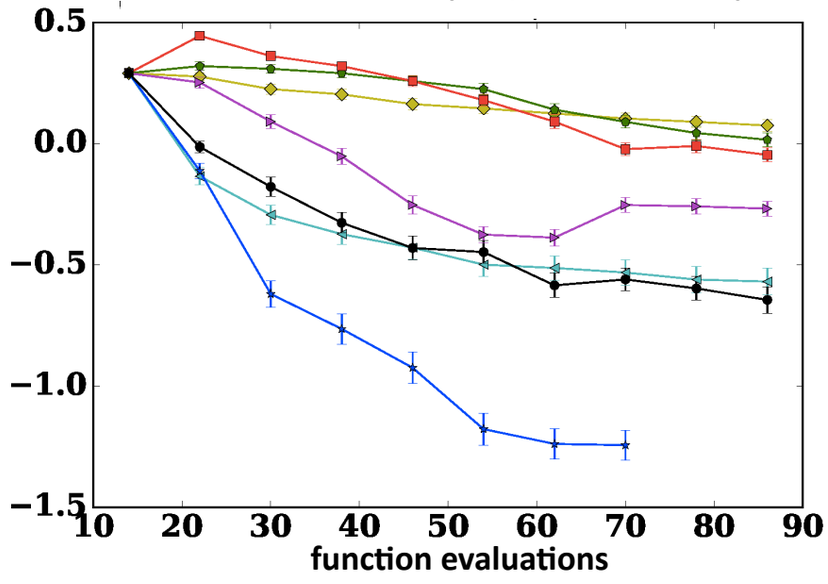
KG provides substantial value over EI when we have gradients

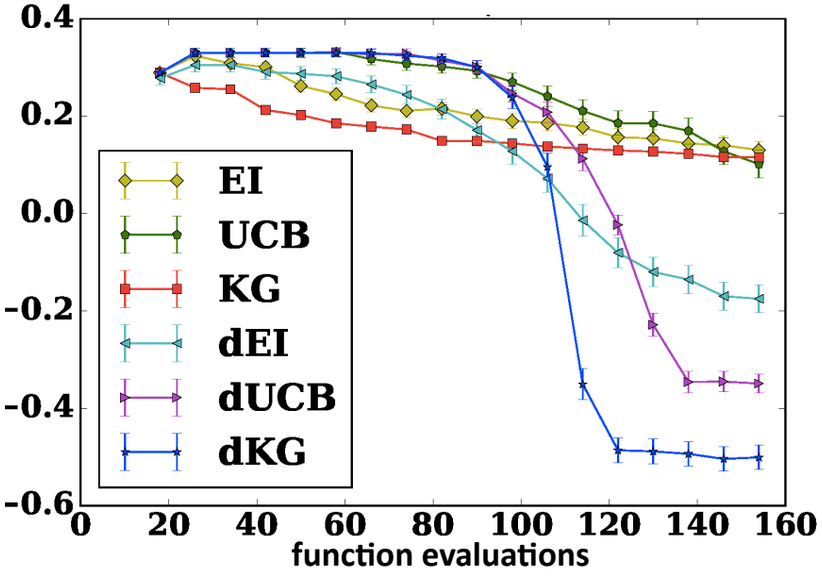






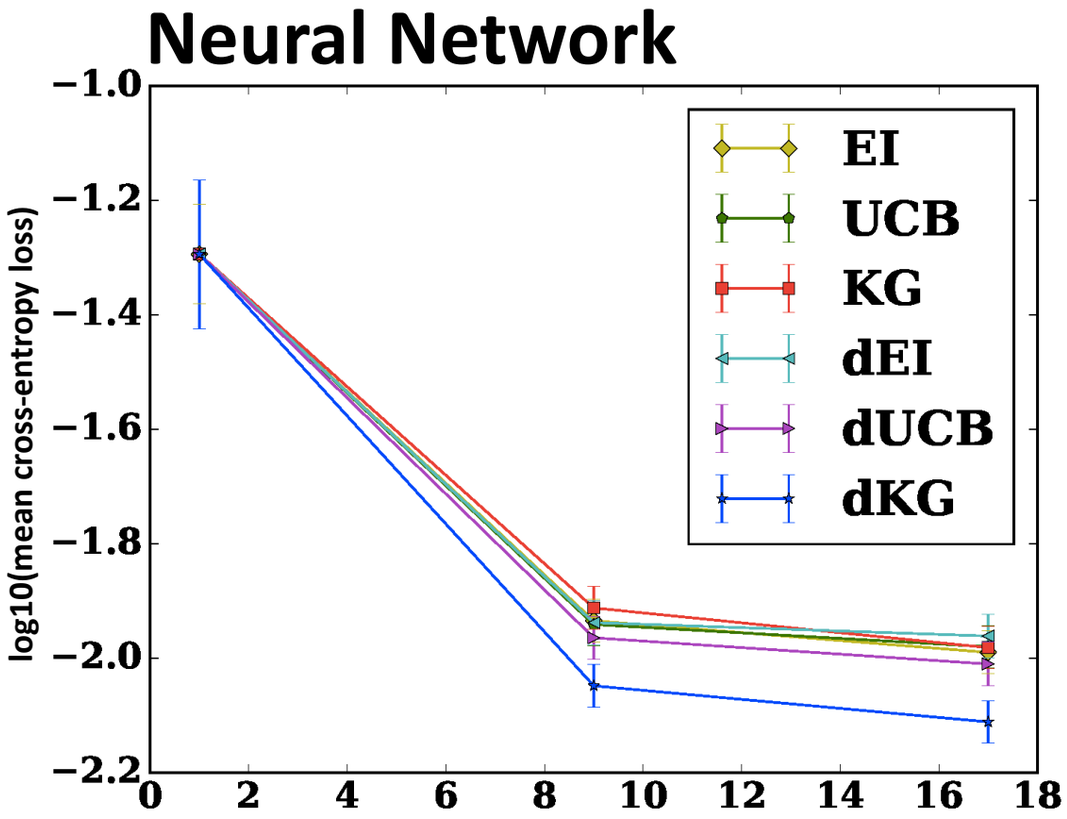


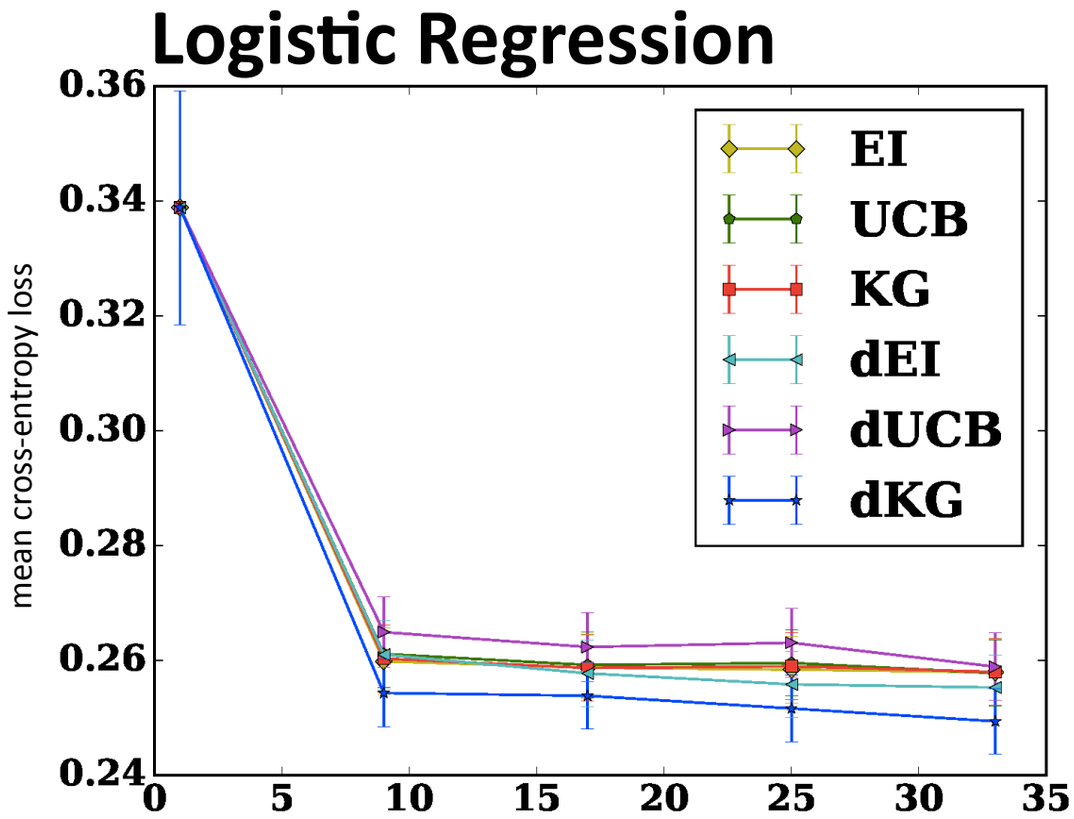


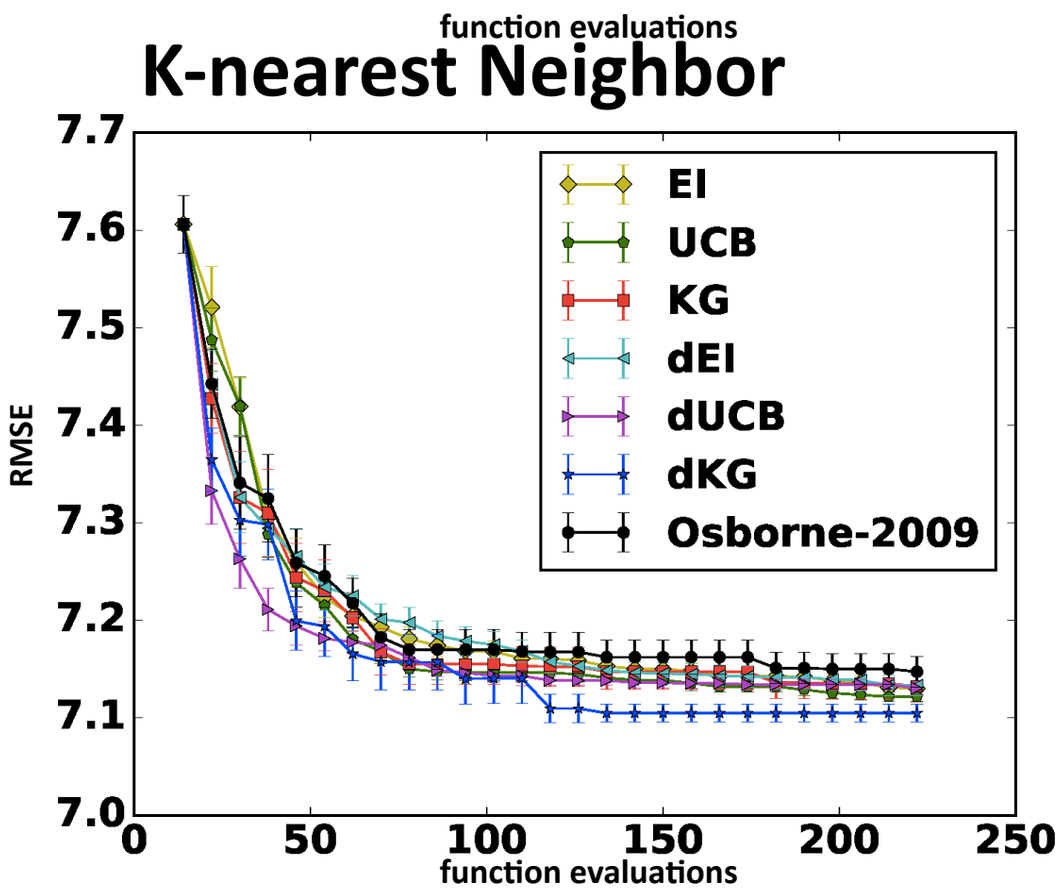


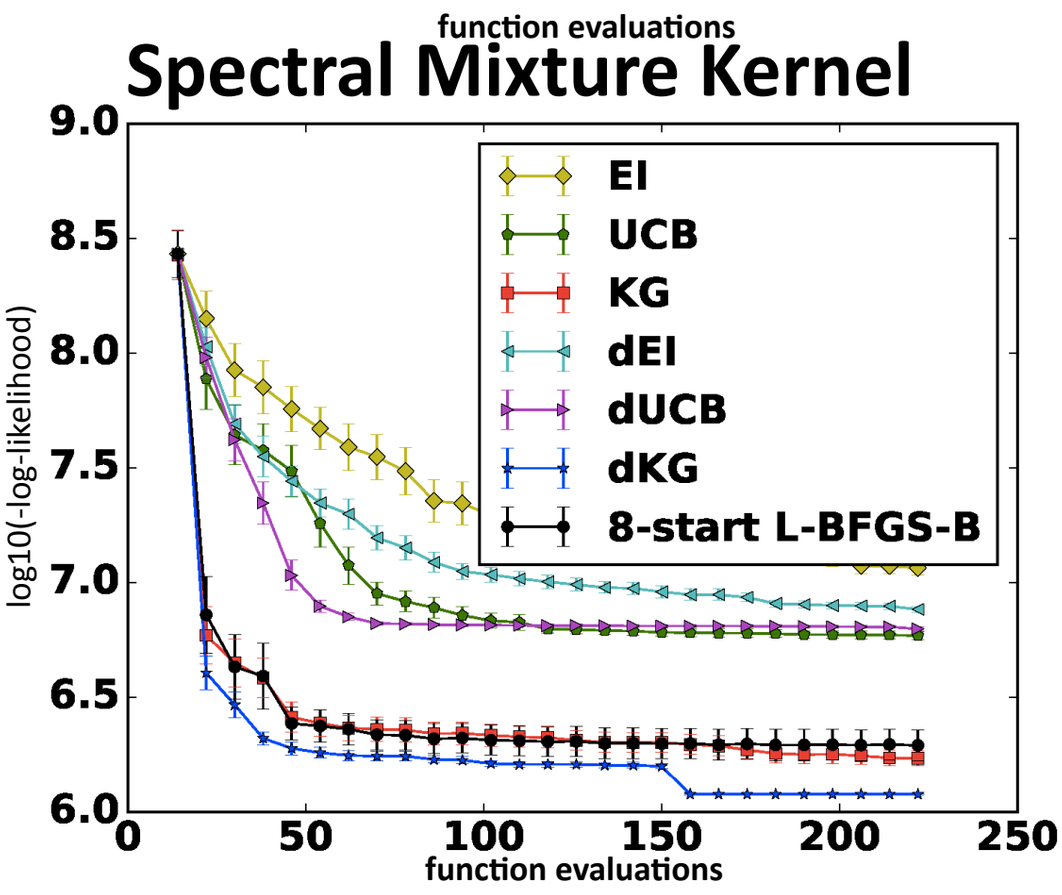
Wu, Poloczek, Wilson & F., Bayesian Optimization with Gradients, NIPS'17

KG provides substantial value over EI when we have gradients





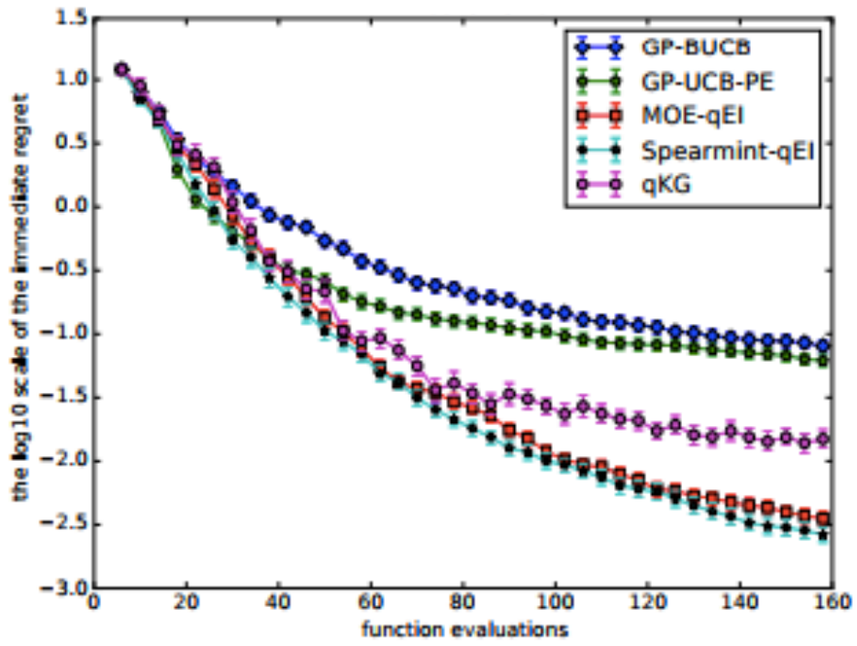




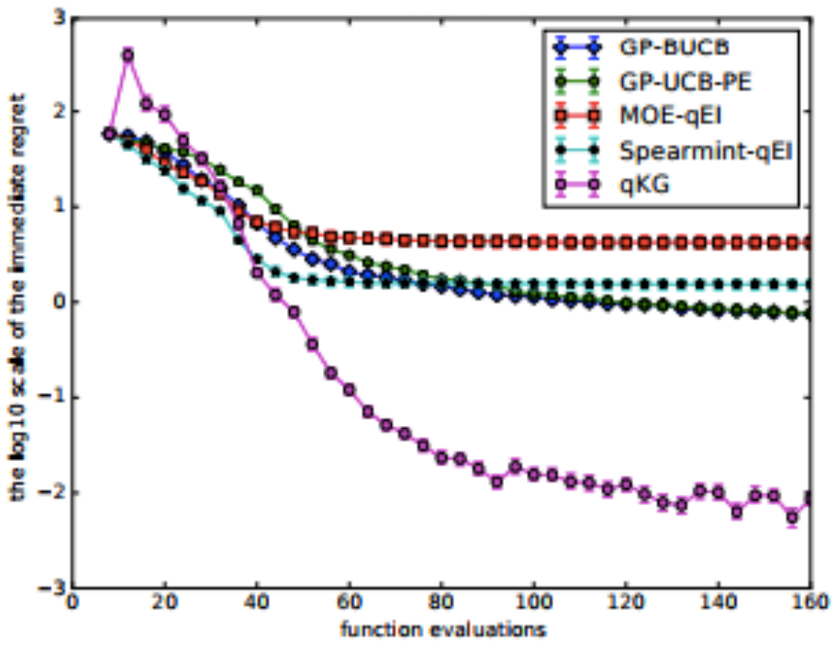
Wu, Poloczek, Wilson & F., Bayesian Optimization with Gradients, NIPS'17

KG provides substantial value over EI when we **don't have gradients** when there is noise, or we are in > 1 dim.

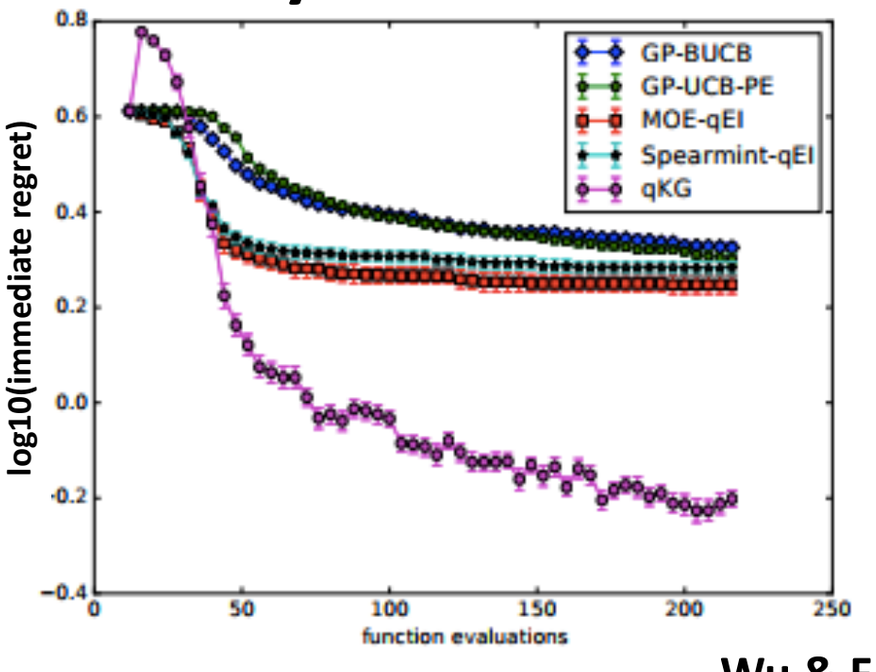
Branin (2-d, batch size=4, no noise)



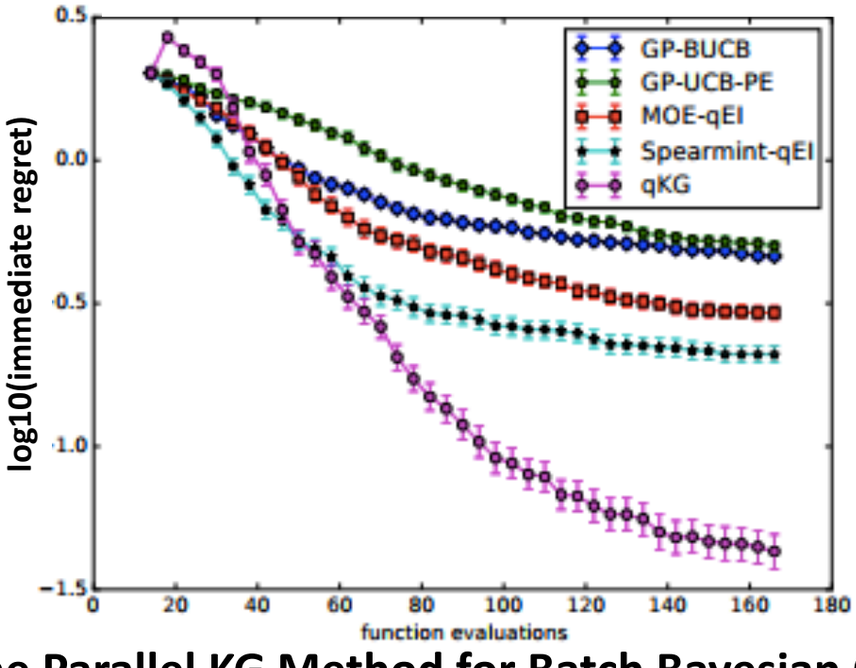
Rosenbrock (3-d, batch size=4, no noise)



Ackley (5-d, batch size=4, no noise)



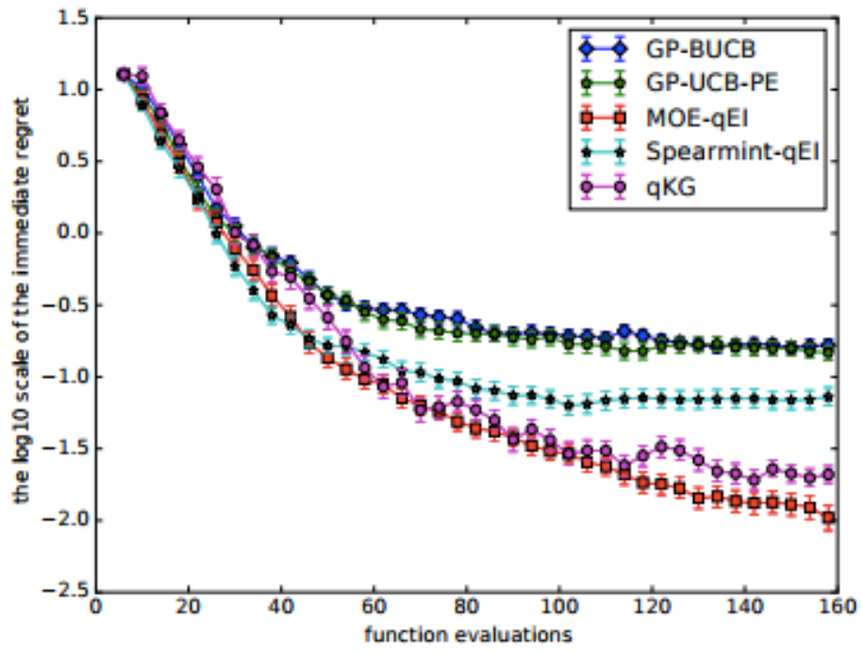
Hartmann (6-d, batch size=4, no noise)



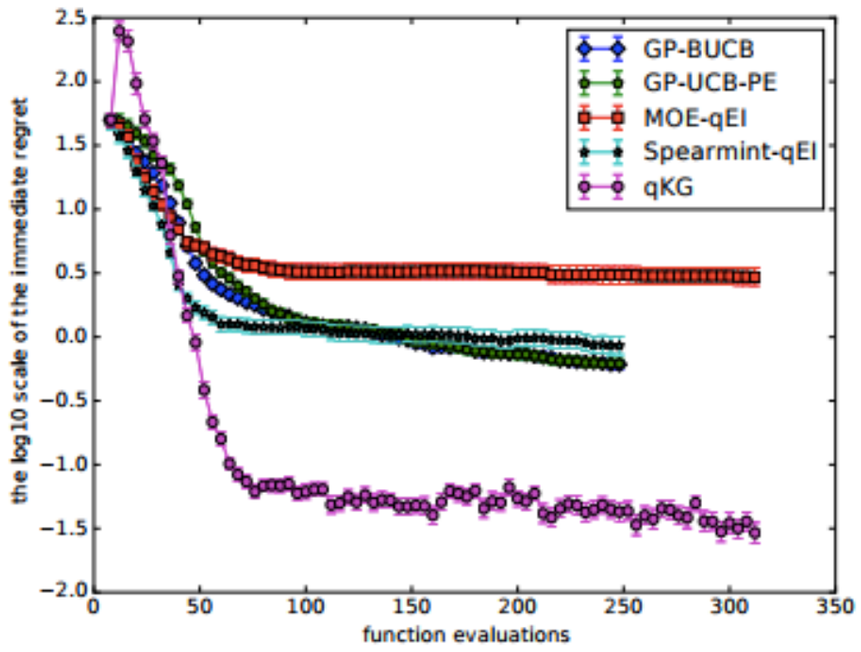
Wu & F., The Parallel KG Method for Batch Bayesian Optimization, NIPS'16

KG provides substantial value over EI when we **don't have gradients** when there is noise, or we are in > 1 dim.

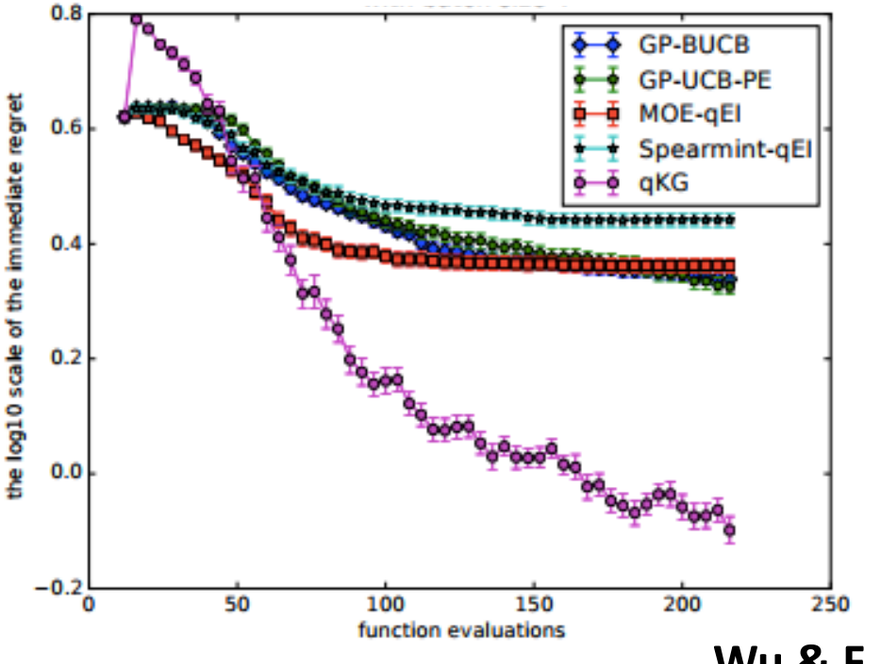
Branin (2-d, batch size=4, noisy)



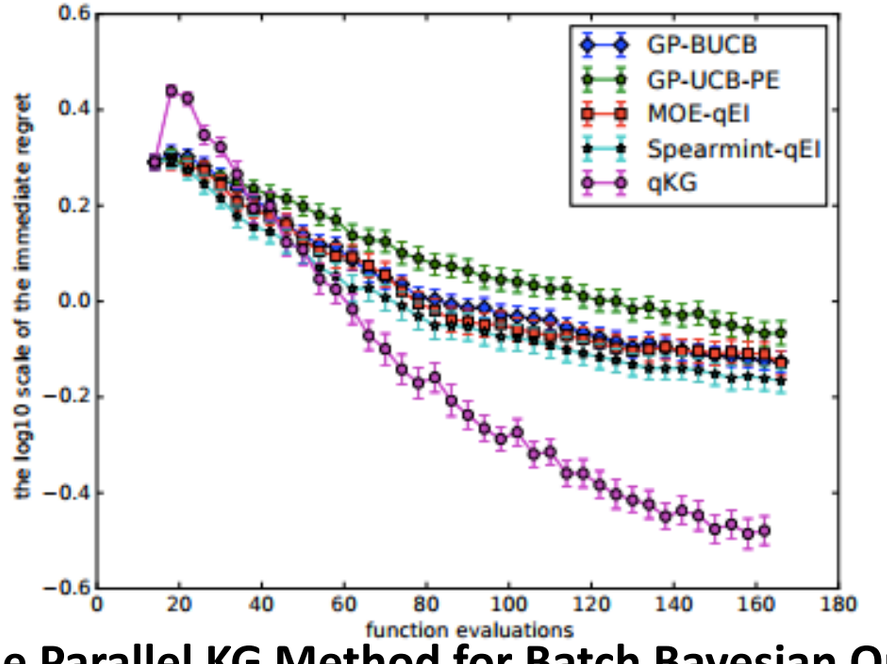
Rosenbrock (3-d, batch size=4, noisy)



Ackley (5-d, batch size=4, noisy)

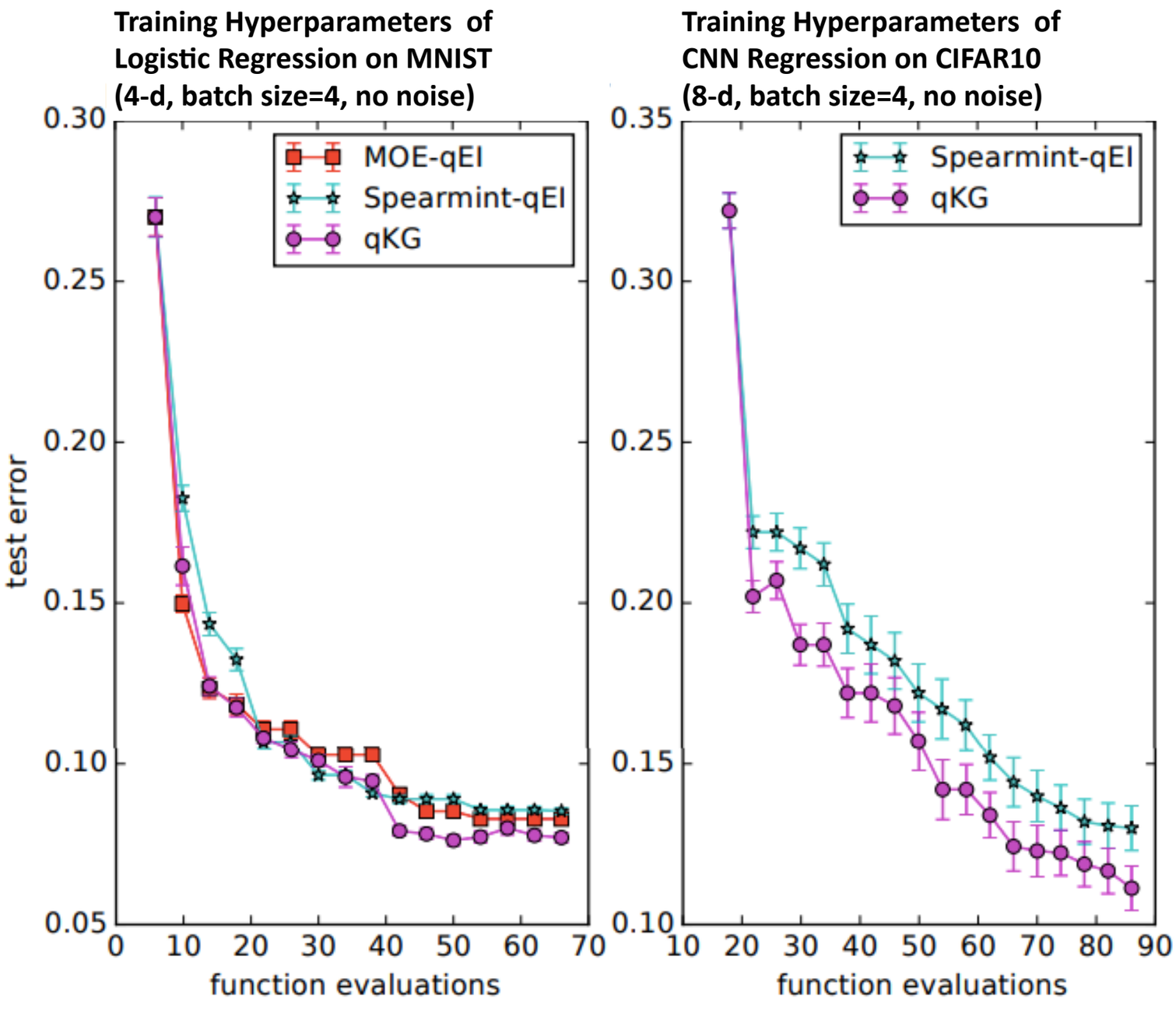


Hartmann (5-d, batch size=4, noisy)



Wu & F., The Parallel KG Method for Batch Bayesian Optimization, NIPS'16

KG provides substantial value over EI when we **don't have gradients** when there is noise, or we are in > 1 dim.



Wu & F., The Parallel KG Method for Batch Bayesian Optimization, NIPS'16

How can we optimize KG(x) efficiently?

Recall this method for computing the KG acquisition function, KG(x)

For i in 1:replications

- Simulate f(x), ∇f(x) from the posterior
- Calculate $\mu^*_{n+1} = \min_{x'} \mu_{n+1}(x')$ from sim'd f(x), ∇f(x)
- Calculate $\mu^*_{n+1} - \mu^*_n$

KG(x) is the average of the simulated $\mu^*_{n+1} - \mu^*_n$ values

Our approach to maximizing $\mathrm{dKG}(\mathrm{x}_{1:q})$

1. Estimate $\nabla \mathrm{dKG}(\mathrm{x}_{1:q})$
   using infinitesimal perturbation analysis (IPA).

2. Use multistart stochastic gradient ascent to find an approximate solution to solve argmax $\mathrm{dKG}(\mathrm{x}_{1:q})$.

Here's how we estimate ∇dKG

- $Y=[f(x_i), \nabla f(x_i) : i=1:q\ ]'$ is multivariate normal with dim. $q(d+1)$
- $Y=m(x_{1:q}) + C(x_{1:q})Z$, where Z is a standard normal random vector
- Write the dependence of $\mu_{n+q}(x)$ on $x_{1:q}$ and Y explicitly as $\mu_{n+q}(x; x_{1:q}, m(x_{1:q}) + C(x_{1:q})Z)$
- $\nabla dKG(x_{1:q}) = \nabla E[\min_{x \text{ in } A}\ \mu_{n+q}(x; x_{1:q}, m(x_{1:q}) + C(x_{1:q})Z)]$
  $= E[\nabla \min_{x \text{ in } A}\ \mu_{n+q}(x; x_{1:q}, m(x_{1:q}) + C(x_{1:q})Z)]$
  $= E[\nabla \mu_{n+q}(x^*; x_{1:q}, m(x_{1:q}) + C(x_{1:q})Z)],$

  where $x^* = \text{argmin}_{x \text{ in } A}\, \mu_{n+q}(x; x_{1:q}, m(x_{1:q}) + C(x_{1:q})Z)$
  and its dependence on $x_{1:q}$ is ignored when taking the gradient.

Here's how we estimate ∇dKG

1. Calculate the mean $m(x_{1:q})$ and the Cholesky decomposition $C(x_{1:q})$ of the covariance matrix of $Y=[f(x_i), \nabla f(x_i) : i=1:q\ ]'$ under the time n posterior distribution.
2. Simulate a standard normal random vector Z. Let $Y = m(x_{1:q}) + C(x_{1:q})Z$
3. Use a nonlinear solver to calculate x^* in $\operatorname{argmin}_x\ \mu_{n+q}(x; x_{1:q}, Y)$
4. Our estimator of $\nabla KG(x_{1:q})$ is $G=\nabla\mu_{n+q}(x^*; x_{1:q}, m(x_{1:q}) + C(x_{1:q})Z)$, holding x^* fixed

Our estimator of the gradient is unbiased

Theorem: When the posterior mean μ_n and covariance kernel Σ_n are continuously differentiable and the domain A is compact,
G is an unbiased estimator of $\nabla dKG(x_{1:q})$

Proof:

- Use conditions in L'Ecuyer 1990 to interchange ∇ and expectation.
- Use the envelope theorem (Milgrom and Segal 2002) to hold x^* fixed

KG converges to a globally optimal solution over a discrete domain

Theorem: When the domain A is discrete & finite, the KG algorithm is consistent, i.e.,

$$\lim_{N\to\infty} f(x^*(\mathrm{dKG}, N)) = \max_{x\in A} f(x)$$

almost surely under the prior, where *x*(KG,N)* is the solution computed by KG after N batches of samples.

Wu, Poloczek, Wilson & F., Bayesian Optimization with Gradients, NIPS'17

KG is useful for other BO problems too: multi-task, multi-fidelity & multi-information source optimization

Poloczek, Wang & F., Multi Information-Source Optimization, NIPS'17

KG is useful for other BO problems too: multi-task, multi-fidelity & multi-information source optimization

- objective f(s,x) ~ GP(mu,Sigma)
- cost c(s,x) ~ GP(mu,Sigma)
- s indexes information sources (IS)
  s=0 is the target we wish to optimize
  x indexes designs
- Goal: solve $\min_x f(0,x)$ using (possibly noisy) queries to f(s,x) at cost c(s,x)

Poloczek, Wang & F., Multi Information-Source Optimization, NIPS'17

Expected Improvement isn't that helpful when used directly in this problem

When you sample $s \neq 0$ at x:
you change the posterior on f,
but you don't observe $f(0,x)$.
There is no improvement in the best solution seen

Using EI requires hacking [see, e.g., Lam et al. 2015]

KG works out of the box

KG(s,x) := $E_n[\mu^*_n - \mu^*_{n+1} \mid$ query x using IS s],

where, $\mu^*_n := \min_x \mu_n(x)$

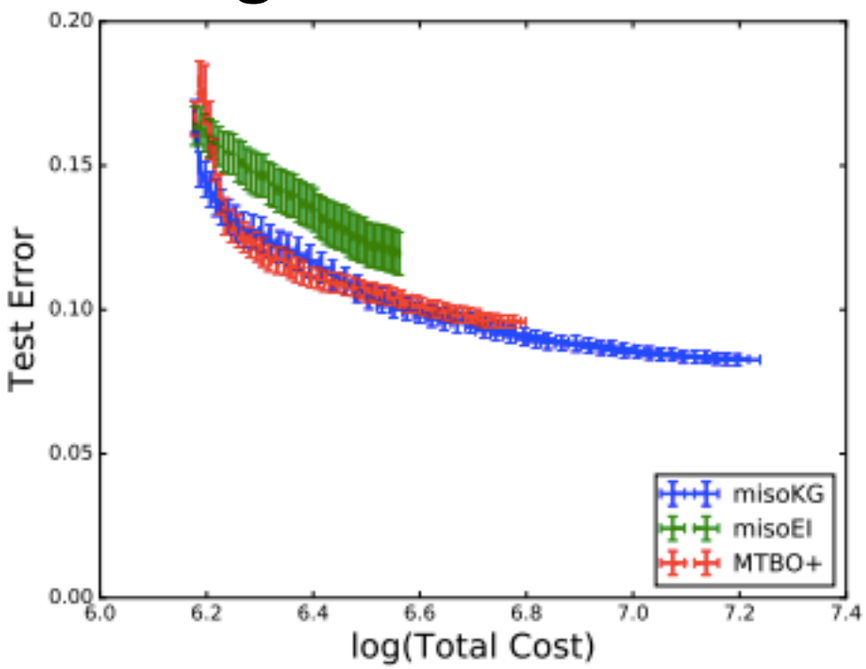
$\mu_n(s,x) := E_n[f(0,x)]$ is the posterior mean on the target at x

Since cost varies, we sample the s,x that maximizes KG(s,x)/cost(s,x)

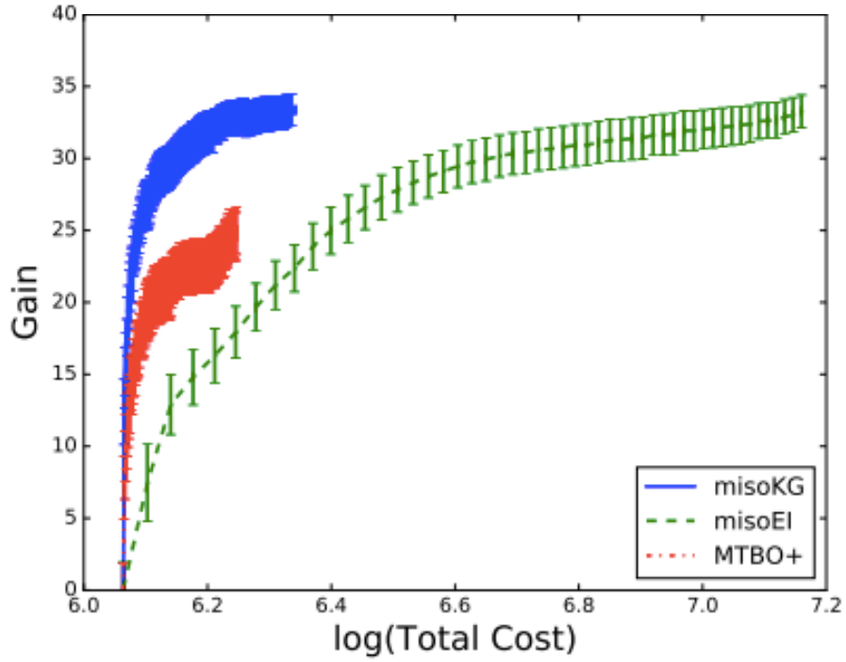
Poloczek, Wang & F., Multi Information-Source Optimization, NIPS'17

KG provides substantial value over EI (and predictive entropy search)

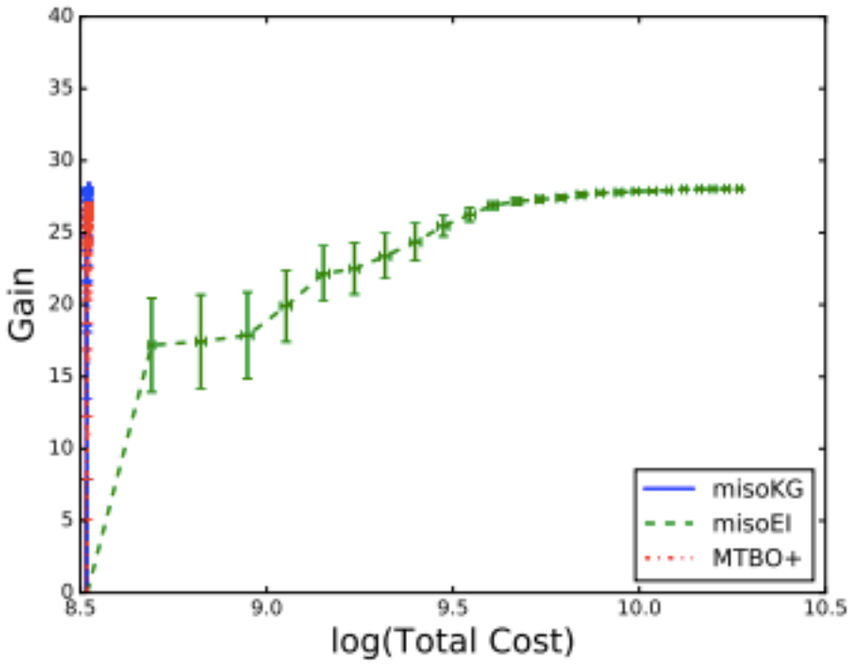
Image Classification [Swersky et al. 2013]



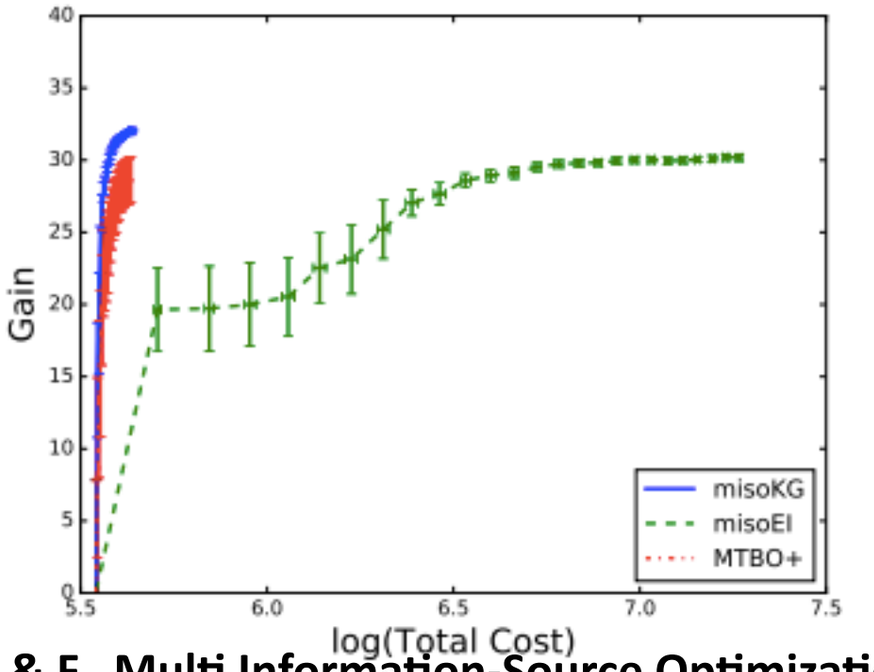
Assemble to Order



Rosenbrock [Lam et al. 2015]



Rosenbrock #2



Poloczek, Wang & F., Multi Information-Source Optimization, NIPS'17

Predictive Entropy Search (PES)

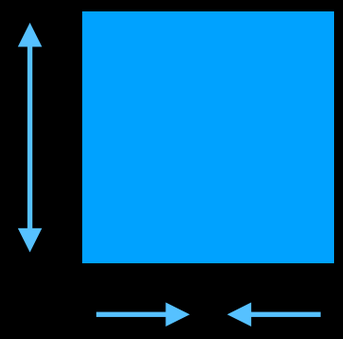
- PES [Hernandez-Lobato et al. 2014] selects the sample that maximizes the expected reduction in the entropy of x*

$$PES(x) = H[p_n(x^*)] - E[H(p_{n+1}(x^*)) \mid \text{query } x]$$

- Works well, if reducing the entropy of x* corresponds to reducing simple regret
- Much more expensive to compute than EI

Reducing the entropy of x* by a lot doesn't always mean you reduce simple regret by a lot

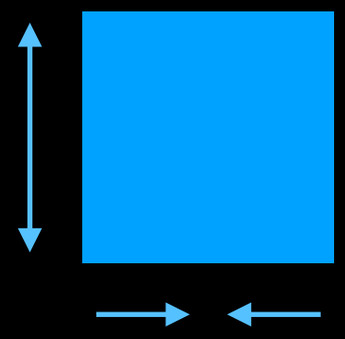
- Our goal: solve $\min_x f(x)$, where x in R^2
- Let $g(x_1,x_2) = f(x_1 / 10, x_2 * 10)$



- When we run PES on g instead of f, PES will work much harder to learn x_2* and will have very different simple regret

Reducing the entropy of x* by a lot doesn't always mean you reduce simple regret by a lot

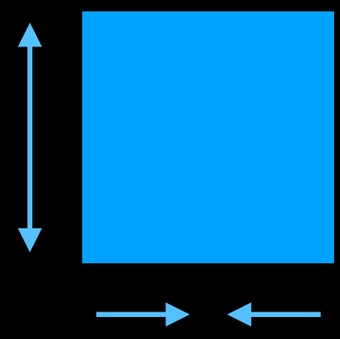
- Our goal: solve $\min_x f(x)$, where x in R^2
- Let $g(x_1,x_2) = f(x_1 / 10, x_2 * 10)$



- When we run PES on g instead of f, PES will work much harder to learn x_2* and will have very different simple regret
- To make the problem worse, suppose f is additive across coordinates, and we observe directional derivatives

Reducing the entropy of x* by a lot doesn't always mean you reduce simple regret by a lot

- Our goal: solve $\min_x f(x)$, where x in R^2
- Let $g(x_1,x_2) = f(x_1 / 10, x_2 * 10)$



- When we run PES on g instead of f, PES will work much harder to learn x_2* and will have very different simple regret
- In contrast, KG's simple regret is invariant to such transformations

Give KG a try

- KG tends to have better query efficiency than EI when the improvement in the posterior is not at the queried point.
- Both KG & PES value improvement in the posterior away from the queried point.
- KG directly values reduction in the simple regret, while PES values it indirectly.
- KG is slower to code & compute than EI. It is comparable to PES.

Code is available:
https://github.com/wujian16/Cornell-MOE (try this 1st)
https://github.com/misokg

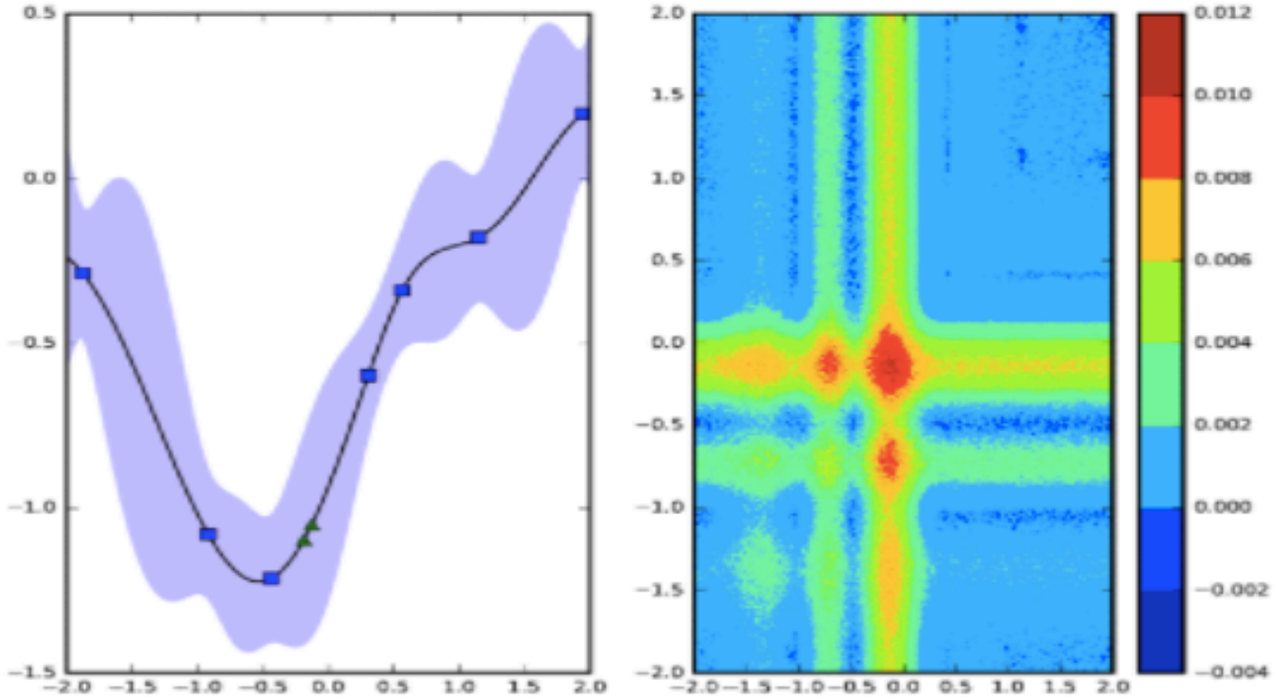
GitHub, Inc. [US] | https://github.com/wujian16/Cornell-MOE/blob/master/README.md

Introduction:

Below we show two demos:

a demo of q-KG on a 1-d derivative-free synthetic function with a batch size q=2.

The left-hand side shows the fitted statistical model and the points suggested by Cornell-MOE. Note that the function evaluation is subject to noise; the right-hand side visualizes the acquisition function according to q-KG criteria.



a demo of d-KG vs. d-EI on a 1-d synthetic function. d-KG explores much more efficiently.

Uber is hiring in BayesOpt:
Email pfrazier@uber.com
or zoubin@uber.com