The Parallel Knowledge Gradient Method for Batch Bayesian Optimization

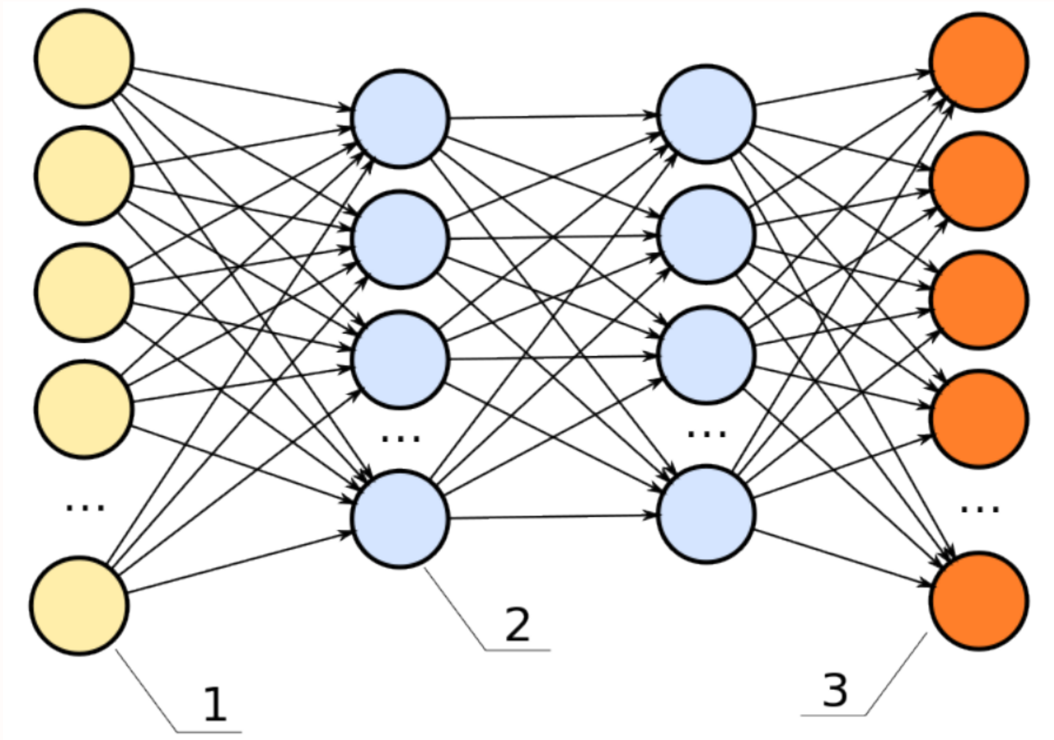
Jian Wu, Ph.D. candidate, Operations Research & Information Engineering, Cornell University
Joint work with Peter I. Frazier

[The Parallel Knowledge Gradient Method for Batch Bayesian Optimization, NIPS 2016.]

We would like to optimize expensive-to-evaluate functions.

- We would like to optimize a function $f : \mathbb{A} \to \mathbb{R}$.
- The feasible set $\mathbb{A}$ is either a box or polyhedron, and compact.
- The unknown function f is smooth.
- We have no information about the derivative of f.
- f is typically expensive to evaluate.
  - Training and testing machine learning algorithms
  - Calibrating parameters of some complex simulators
  - Labor intensive experimental design

Application: tuning machine learning algorithms

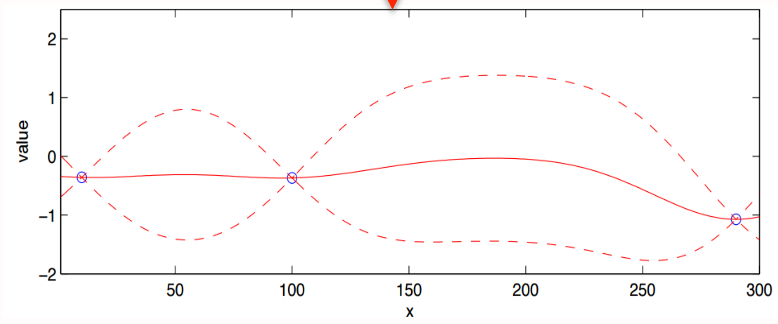


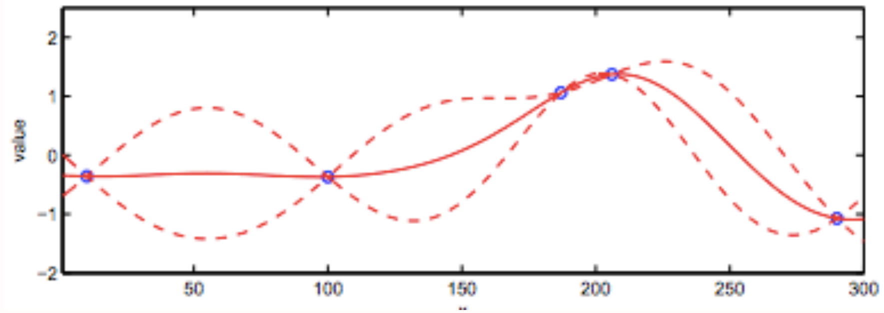
- Number of hidden units each layer
- Learning rate in SGD
- Number of iterations in training
- ……

Batch Bayesian Optimization

- Common BayesOpt algorithms look like:

Update the posterior of the surface





Start from a prior on the surface → Select the next batch of points to evaluate

- We typically use a Gaussian Process prior.
  - Posterior after evaluating n points:

$$f(\mathbf{x}) \sim \mathcal{N}(\mu^{(n)}(\mathbf{x}), K^{(n)}(\mathbf{x}))$$

The Parallel Knowledge Gradient (qKG)

- If we were to stop after n points, $\min_{x \in \mathbb{A}} \mu^{(n)}(x)$ is the minimum of the mean function.

- If we take one additional iteration (q more points), $\min_{x \in \mathbb{A}} \mu^{(n+q)}(x)$ is the minimum of the mean. It depends on where these q points are and their function values.

- The quality of the q points selected is quantified by

$$\min_{x \in \mathbb{A}} \boldsymbol{\mu}^{(n)}(x) - \mathbb{E}_n \left[\min_{x \in \mathbb{A}} \boldsymbol{\mu}^{(n+q)}(x) | \boldsymbol{y}(\boldsymbol{z}^{(1:q)}) \right]$$

The Parallel Knowledge Gradient (qKG)

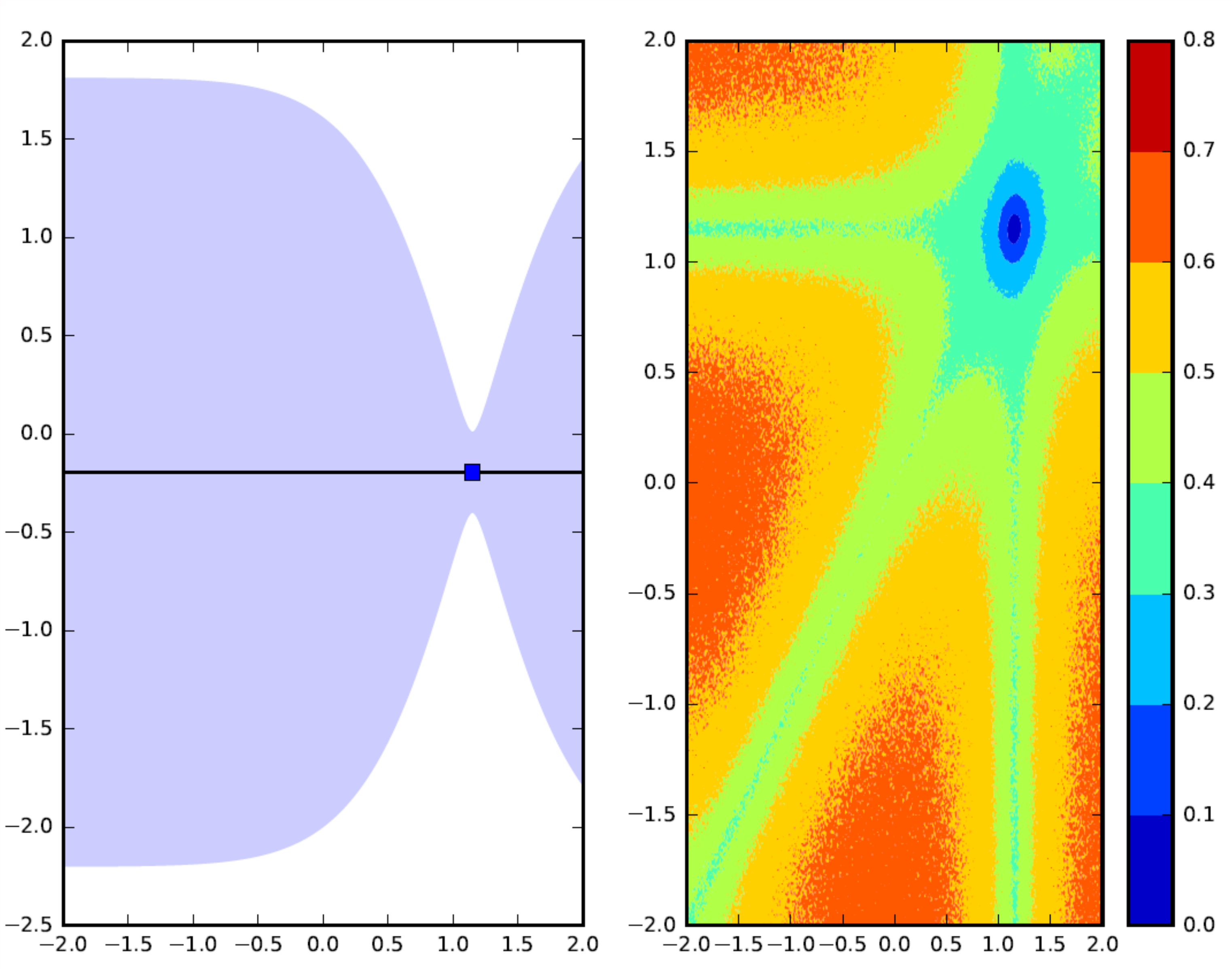
- The q-KG criterion is defined as:

$$q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}) = \min_{x \in \mathbb{A}} \boldsymbol{\mu}^{(n)}(x) - \mathbb{E}_n \left[\min_{x \in \mathbb{A}} \boldsymbol{\mu}^{(n+q)}(x) | \boldsymbol{y}(\boldsymbol{z}^{(1:q)}) \right]$$

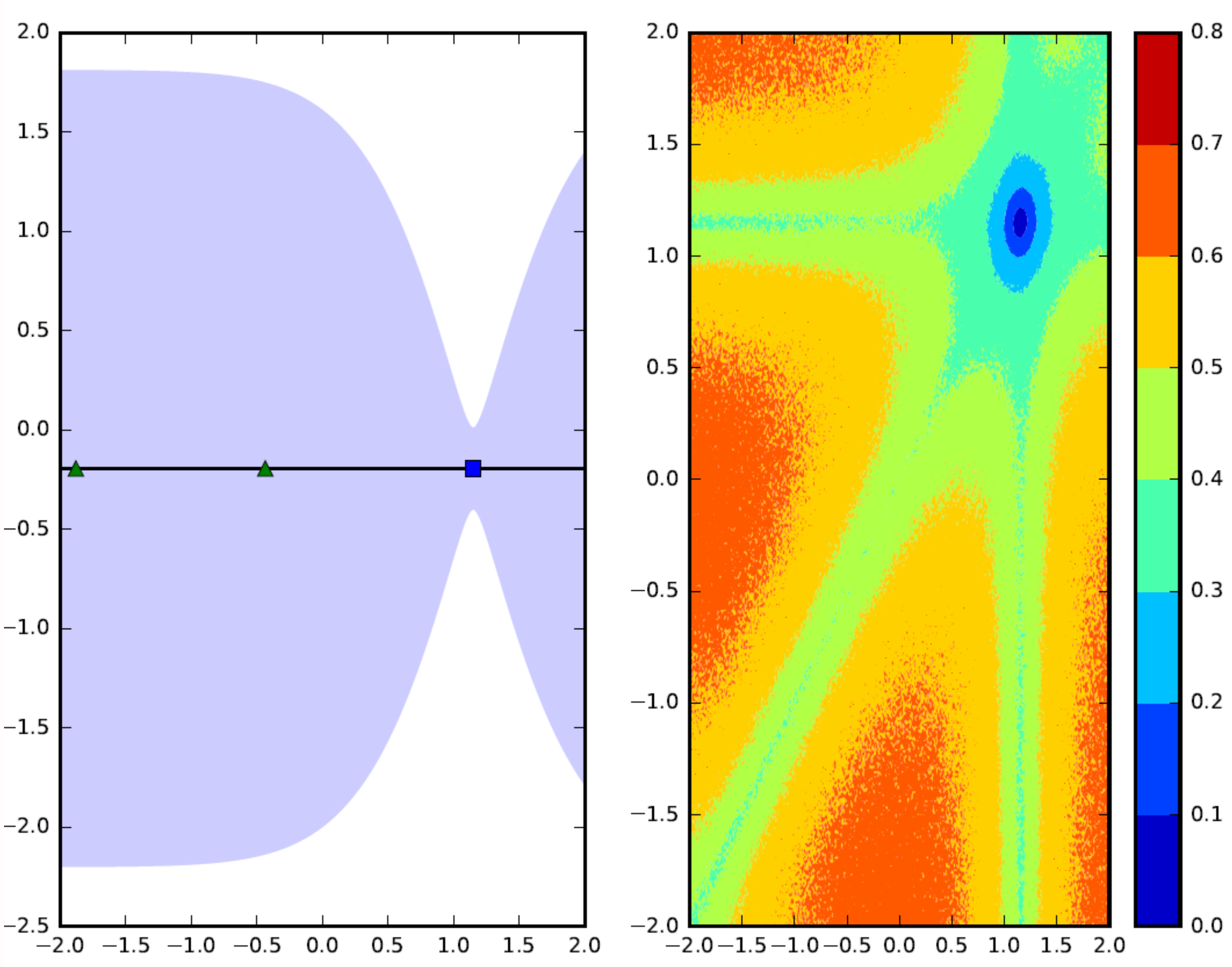
- The q-KG algorithm is to try maximize the criterion above.

$$\max_{\boldsymbol{z}^{(1:q)} \subset \mathbb{A}} q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A})$$

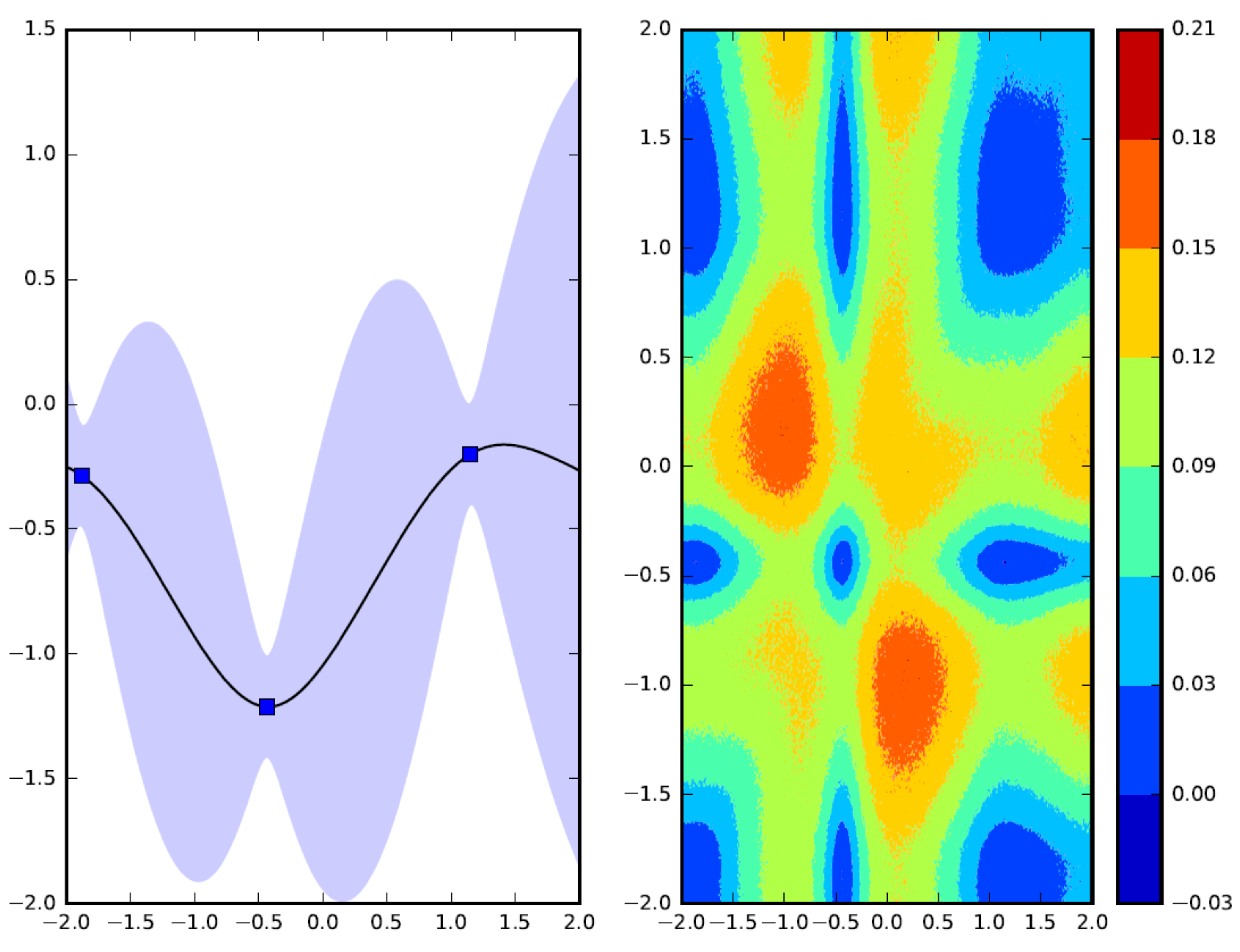
How q-KG works



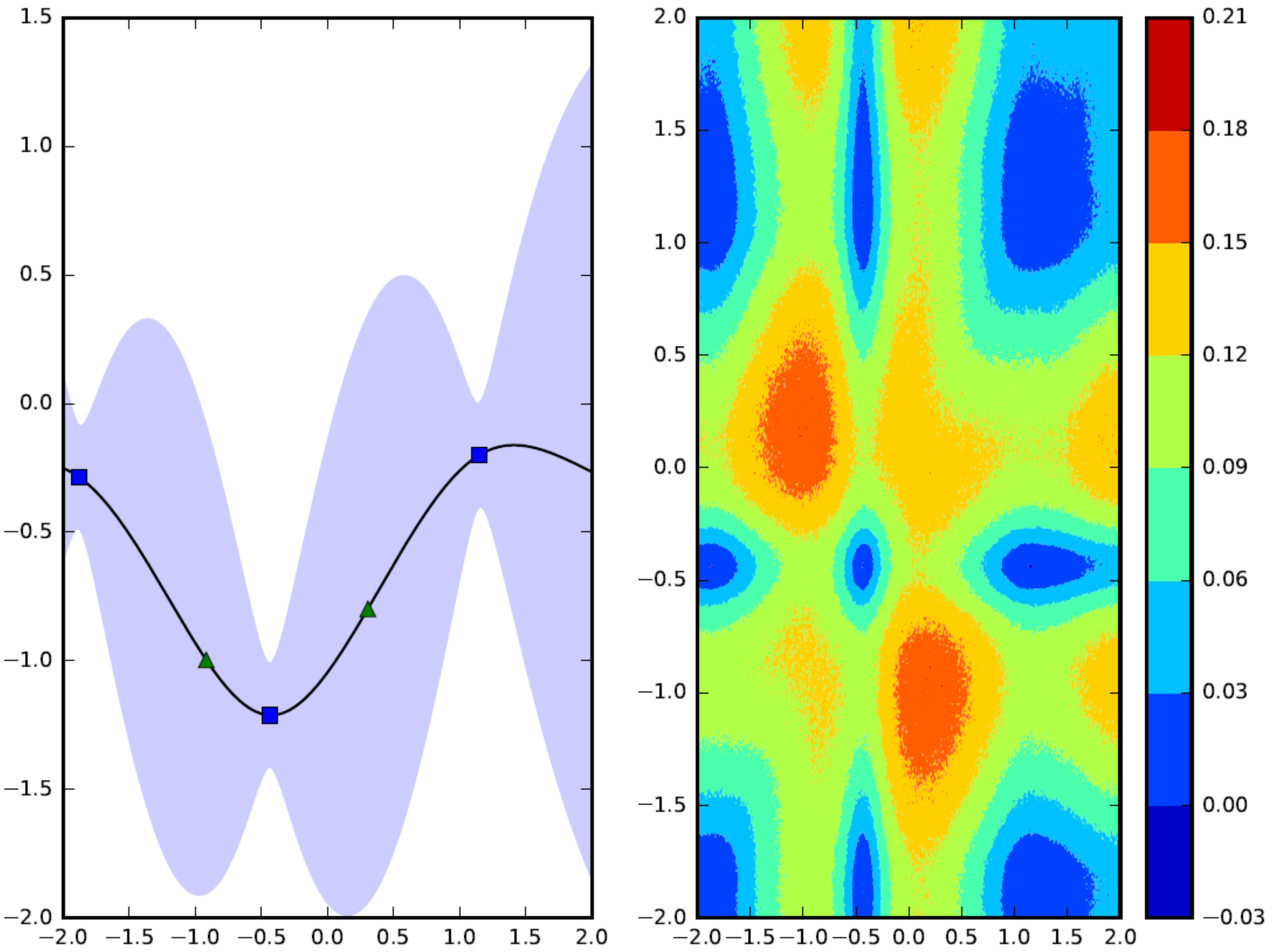
How q-KG works



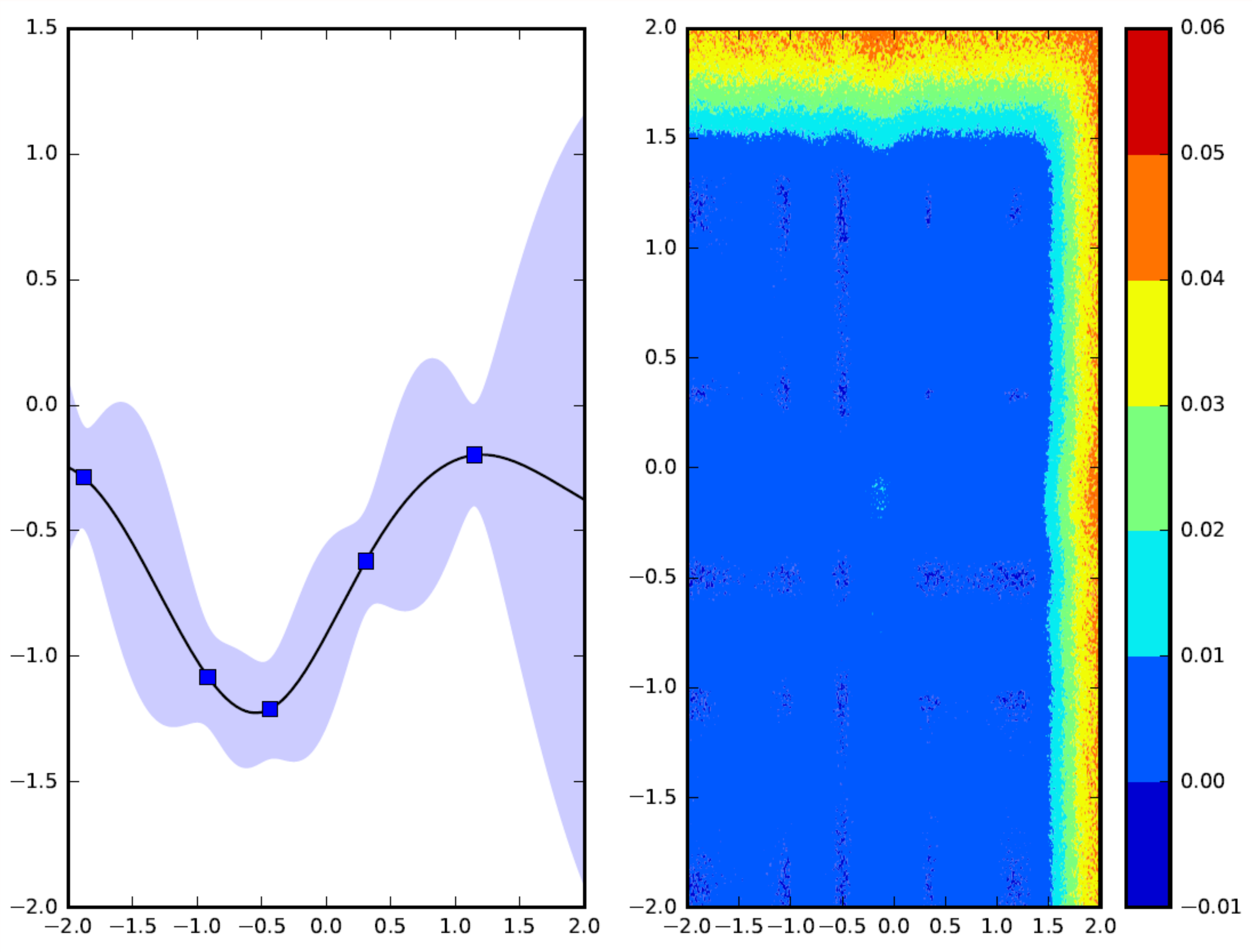
How q-KG works



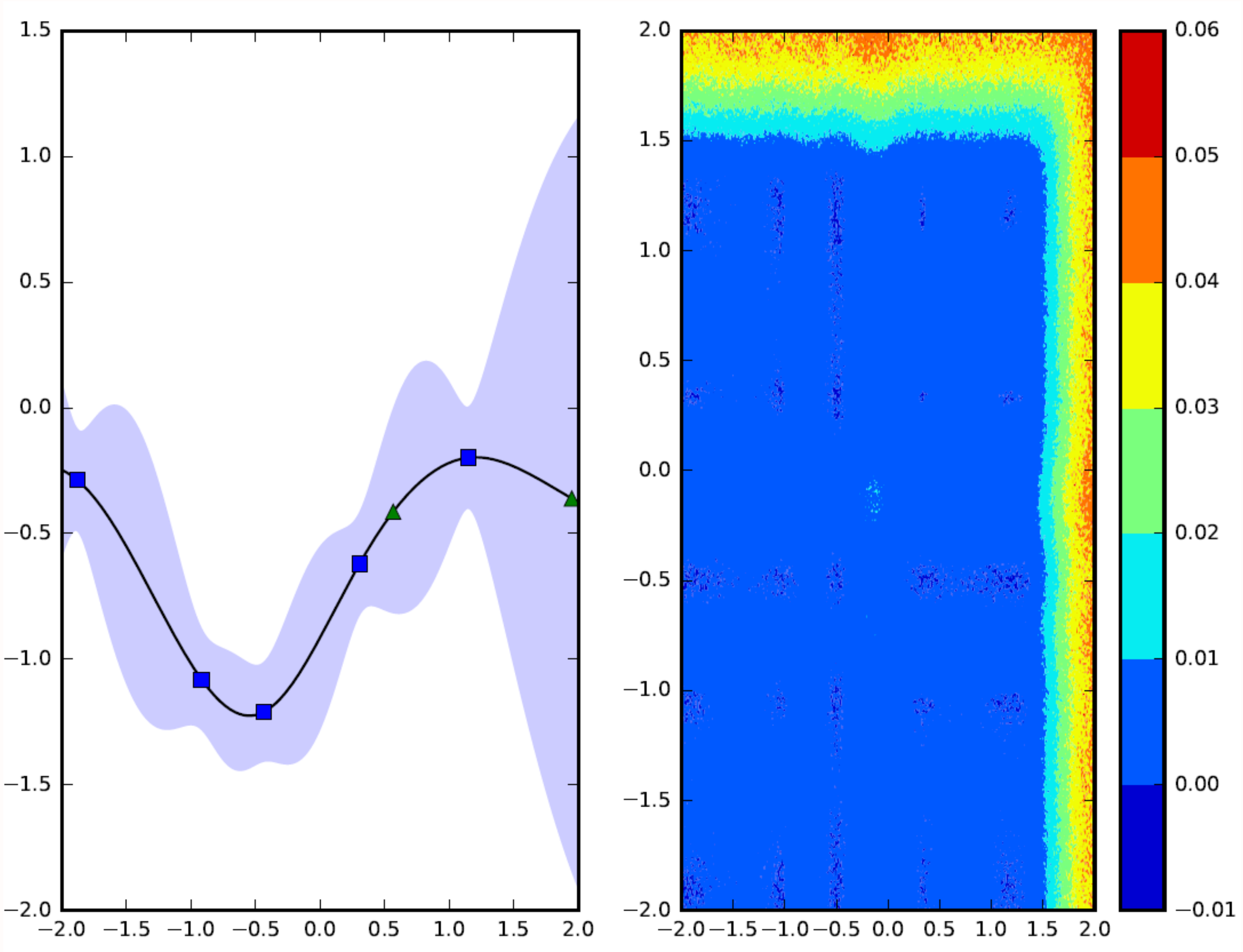
How q-KG works



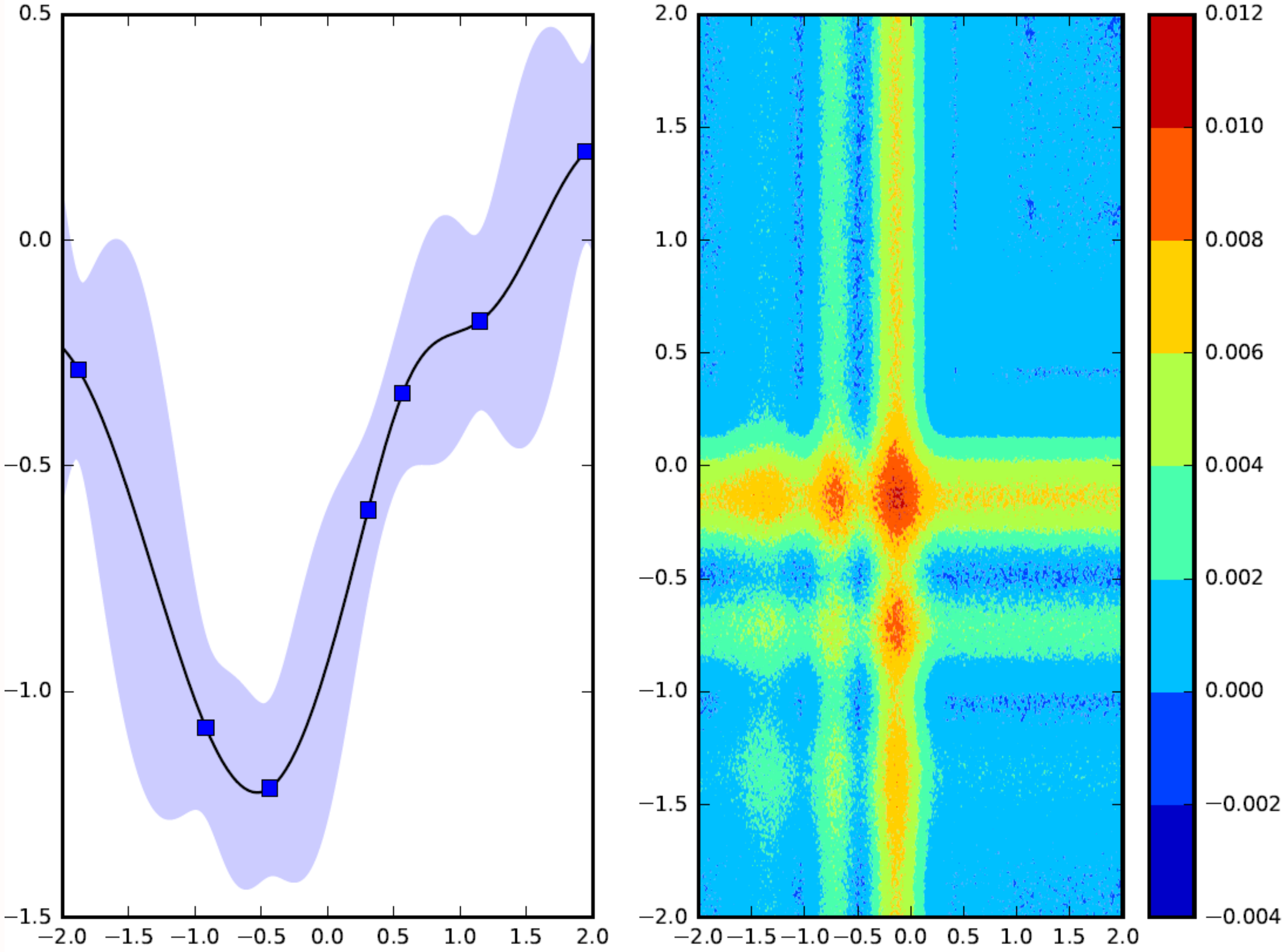
How q-KG works



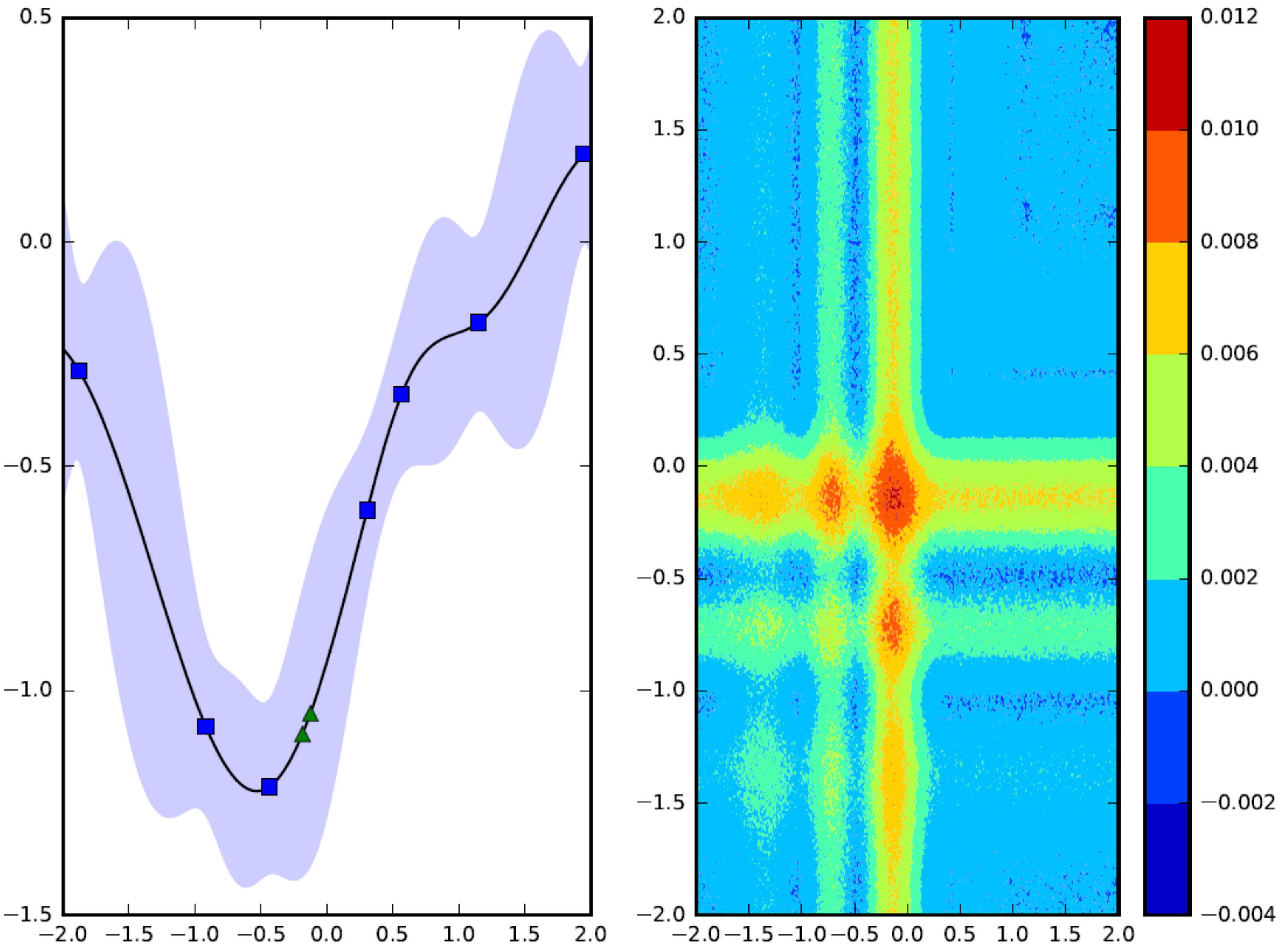
How q-KG works



How q-KG works



How q-KG works



Maximization of q-KG

$$q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}) = \min_{x \in \mathbb{A}} \boldsymbol{\mu}^{(n)}(x) - \mathbb{E}_n \left[\min_{x \in \mathbb{A}} \boldsymbol{\mu}^{(n+q)}(x) | \boldsymbol{y}(\boldsymbol{z}^{(1:q)}) \right]$$

Estimate $\nabla q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A})$

Multi-start Stochastic Gradient Ascent

Estimate the Derivative of q-KG when A is finite

- By Gaussian process properties

$$\boldsymbol{\mu}^{(n+q)}(\mathbb{A}) = \boldsymbol{\mu}^{(n)}(\mathbb{A}) + C^{(n)}(\mathbb{A}, \boldsymbol{z}^{(1:q)})Z_q$$

where $C^{(n)}$ is some function related to posterior covariance function. Z is q-dimensional standard normal.

- q-KG can be rewritten as

$$q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}) = \mathbb{E}_{Z_q}\left[\min \boldsymbol{\mu}^{(n)}(\mathbb{A}) - \min\left(\boldsymbol{\mu}^{(n)}(\mathbb{A}) + C^{(n)}(\mathbb{A}, \boldsymbol{z}^{(1:q)})Z_q\right)\right]$$

- When the prior mean and kernel function is continuously differentiable and A is bounded

$$\nabla q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}) = \mathbb{E}_{Z_q}\left[\nabla\left(\min \boldsymbol{\mu}^{(n)}(\mathbb{A}) - \min\left(\boldsymbol{\mu}^{(n)}(\mathbb{A}) + C^{(n)}(\mathbb{A}, \boldsymbol{z}^{(1:q)})Z_q\right)\right)\right]$$

We discretize A when A is a continuous domain

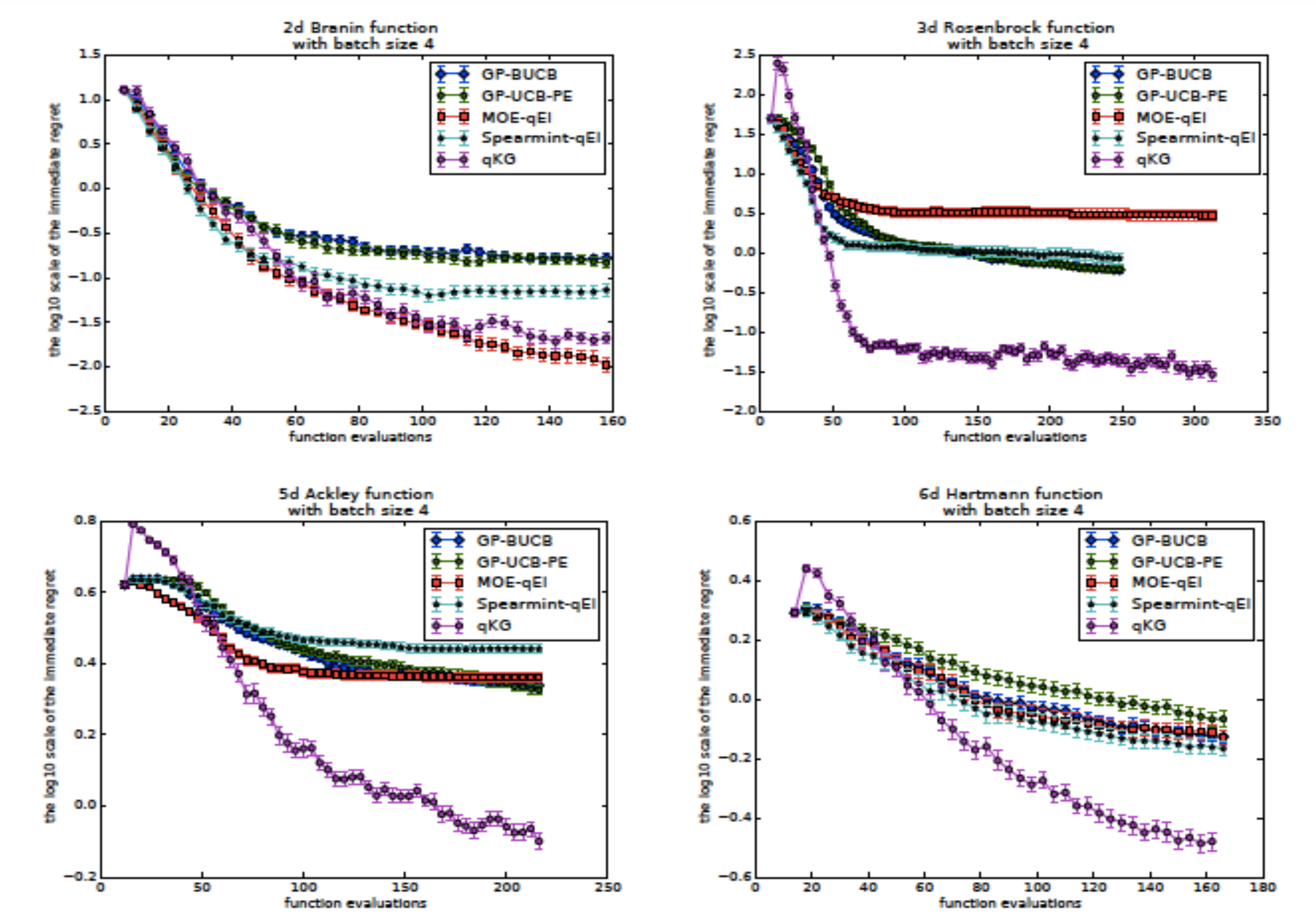
- We approximate A as

$$\mathbb{A}_n = \mathbb{A}_n^M \cup \boldsymbol{x}^{(1:n)} \cup \boldsymbol{z}^{(1:q)}$$

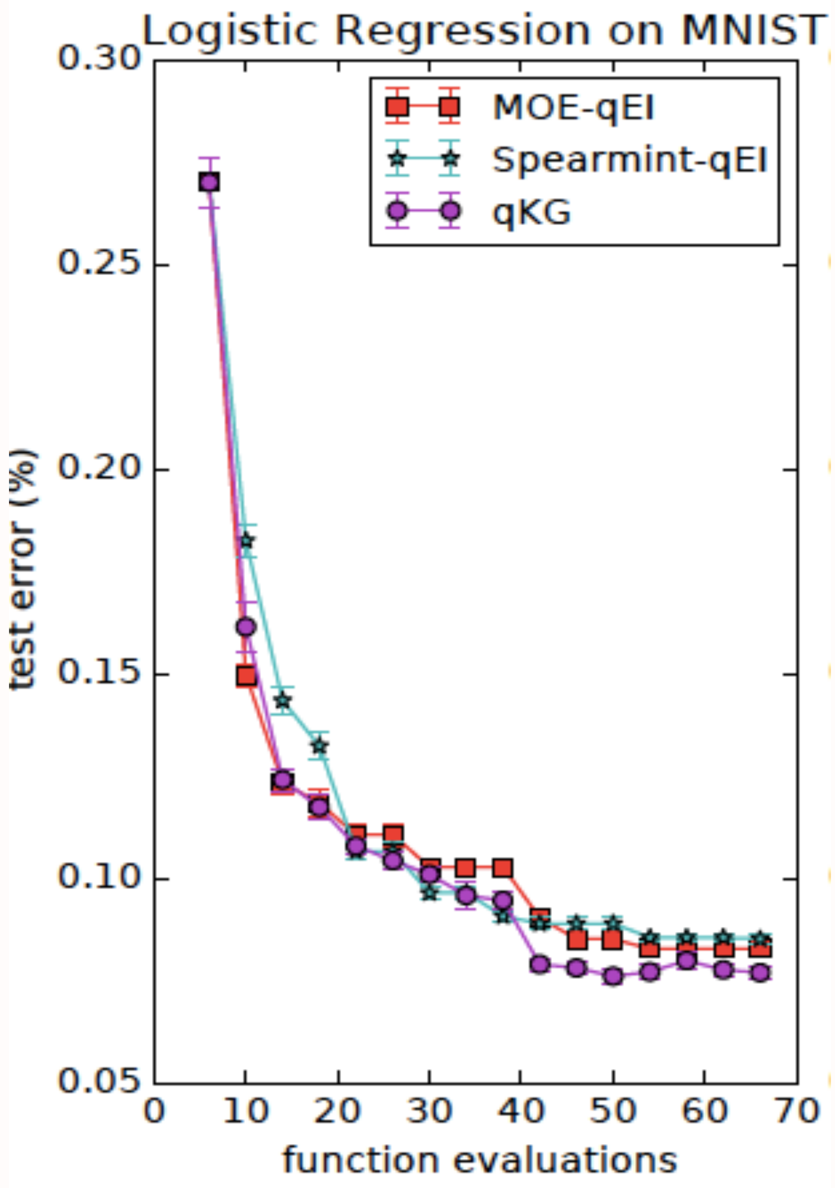
- The $\mathbb{A}_n^M$ is the samples of the global optima based on the current posterior surface: random feature approximation.

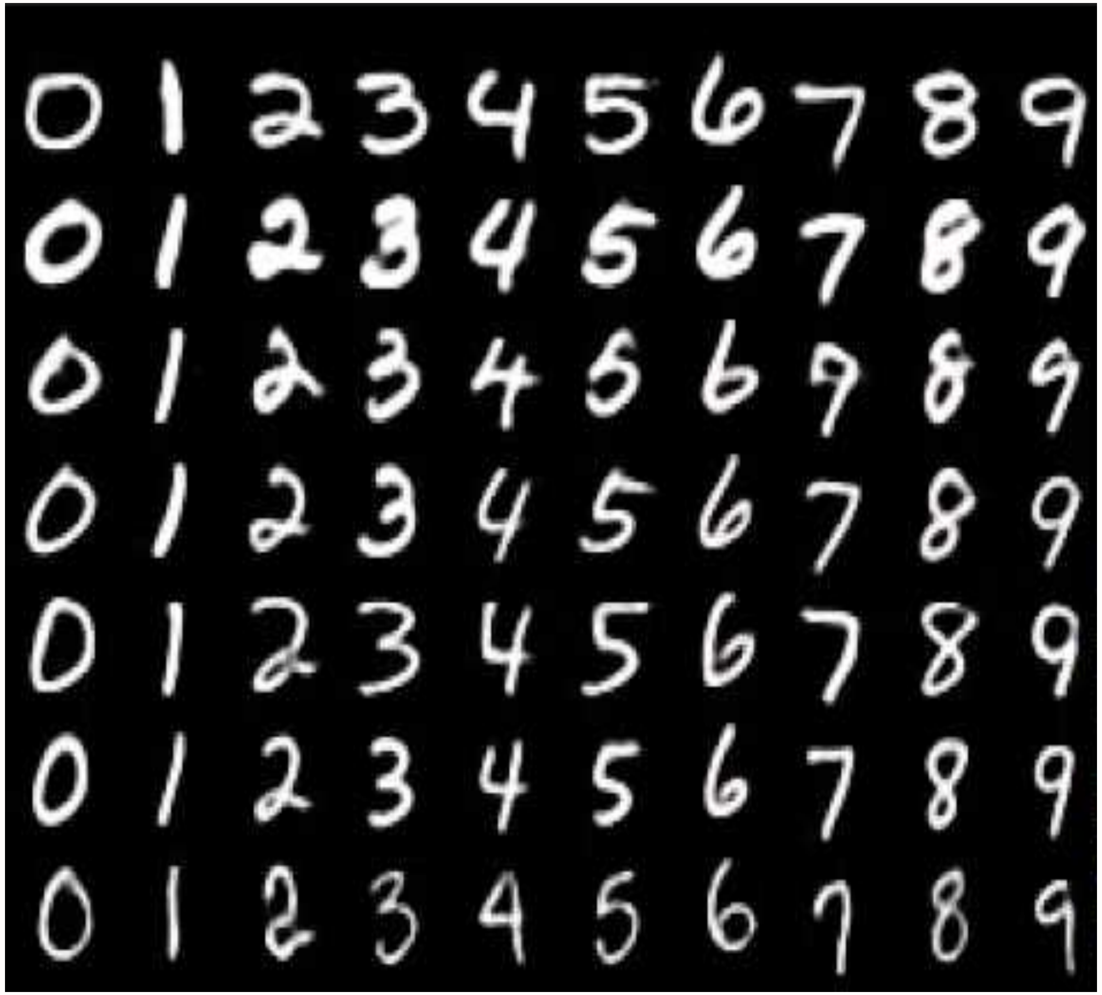
- We then can use the multi-start gradient based optimizer.

q-KG Outperforms Other Algorithms on Noisy Synthetic Functions



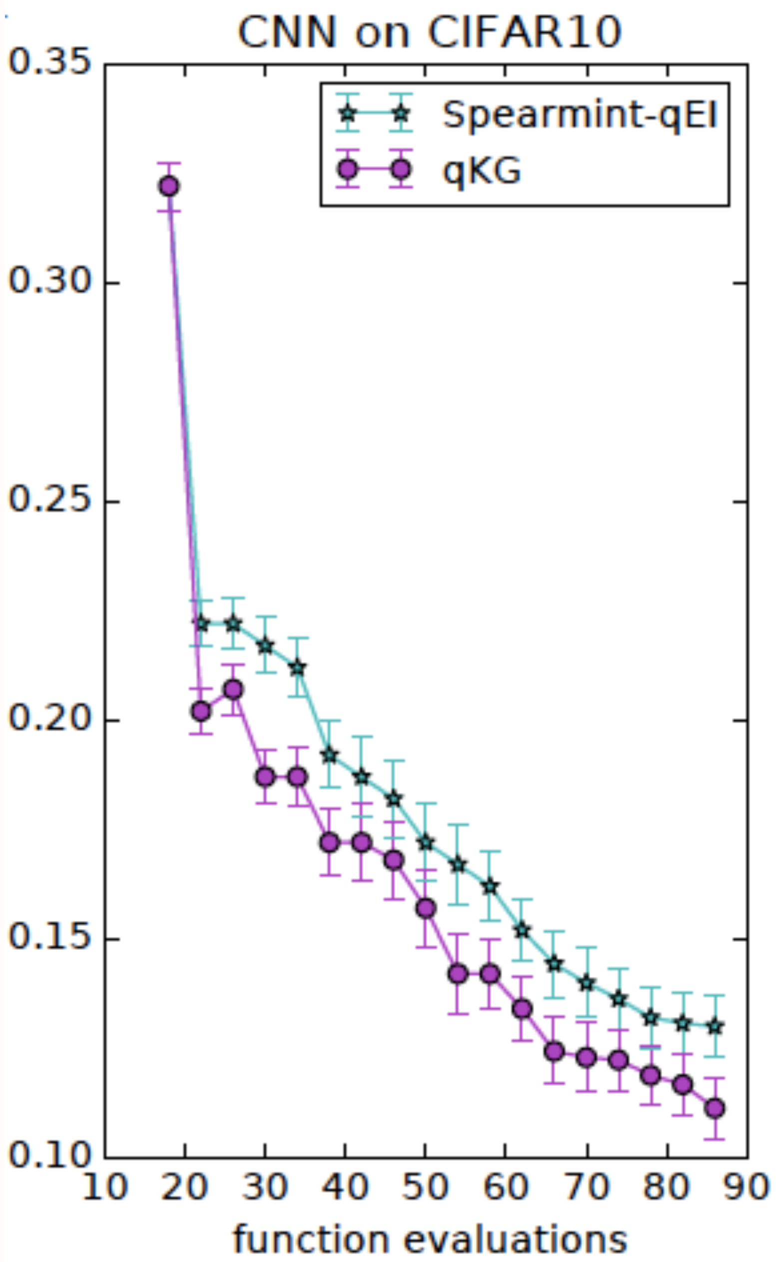
q-KG Outperforms State-of-Art when Tuning Logistic Regression

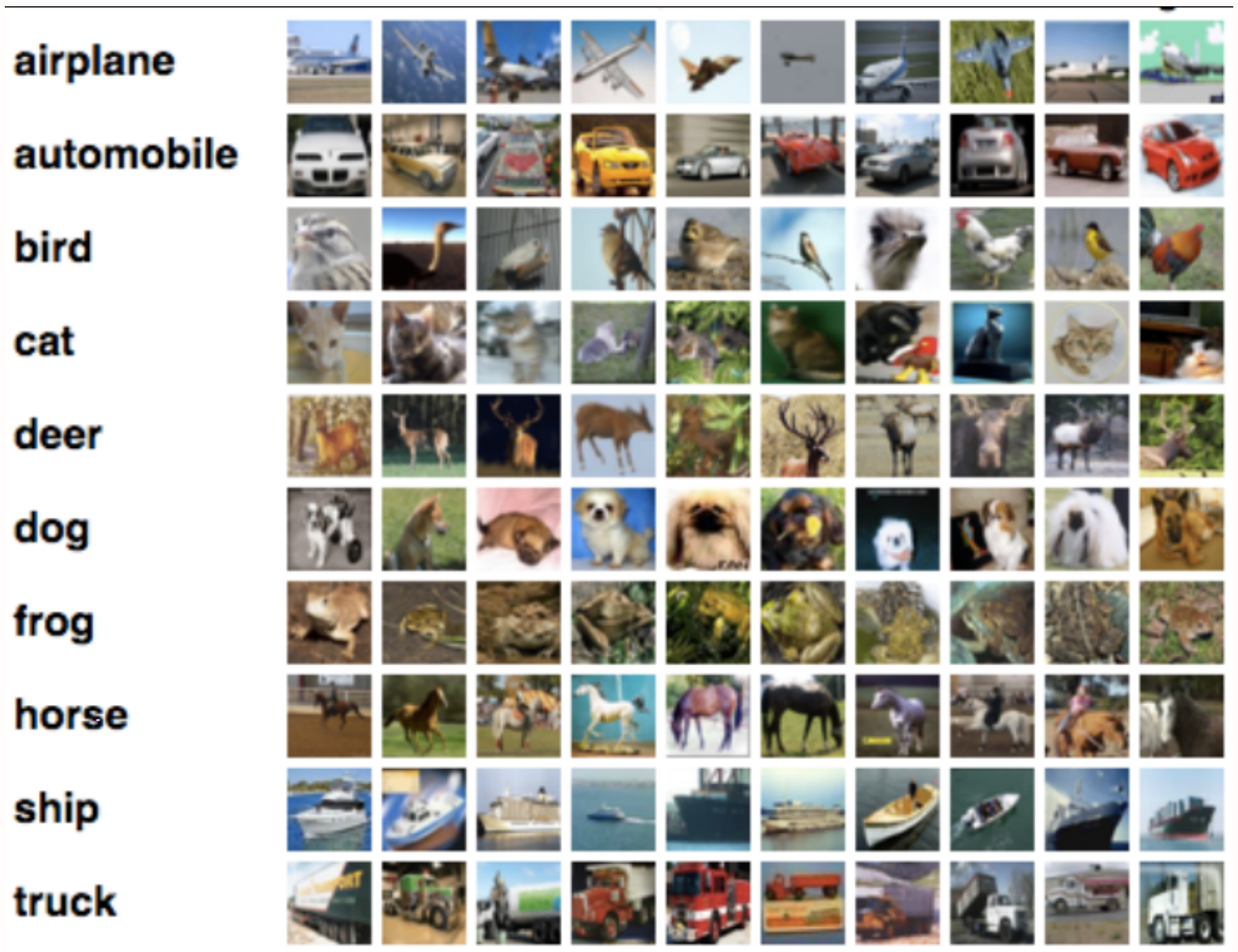




MNIST Dataset, http://yann.lecun.com/exdb/mnist/

q-KG Outperforms State-of-Art when Tuning CNN

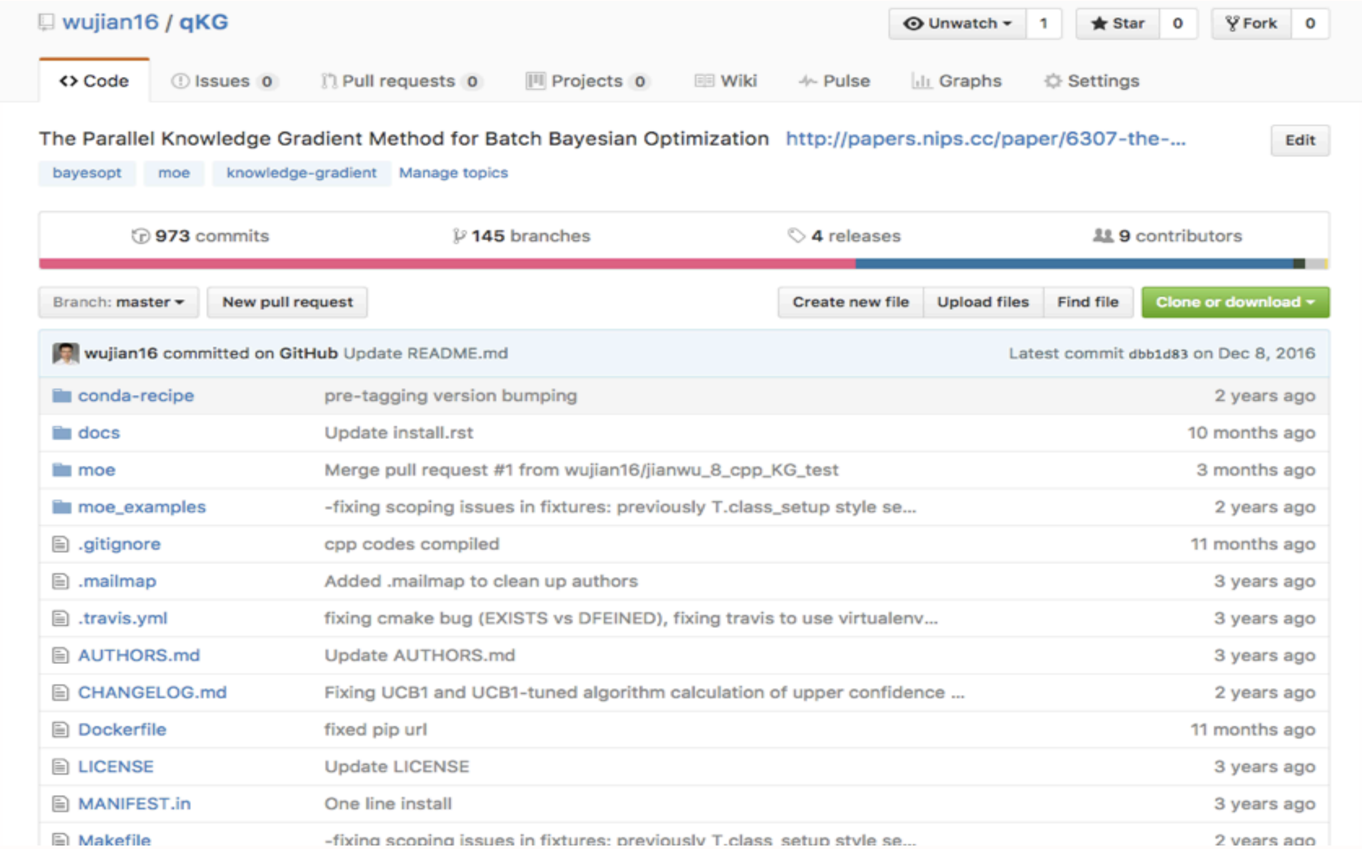




CIFAR10 dataset, https://www.cs.toronto.edu/~kriz/cifar.html

The code is made public, you can use it

- **The paper is published at NIPS 2016 with the same title as this talk.**
- **We build the algorithm into the open-source package: MOE.**
- **The code is completely in C++, with a Python interface.**
- **It is available at https://github.com/wujian16/qKG**



Thanks! Any Question?