Machine Learning
Seoul National University

Homework #(**3**)
JARRY Guillaume

---

1 Q1

Because $f(x)|X, f$ follows a Gaussian process, the probability:

$$P(f(x) > f_n^+|X, f) = 1 - P(f(x) \leq f_n^+|X, f)$$

Then,

$$P(f(x) > f_n^+|X, f) = 1 - P(f(x) \leq f_n^+|X, f)$$

$$P(f(x) > f_n^+|X, f) = 1 - \int_{-\infty}^{f_n^+} \frac{1}{\sqrt{2\pi\sigma_n(x)^2}} exp\left(-\frac{(t-\mu_n(x))^2}{2\sigma_n(x)^2}\right) dt$$

Then, in order to express the probability according to the cdf of the standard gaussian distribution, we need to make the change n variable $s = \frac{t-\mu_n(x))}{\sigma_n(x)}$. And thus:

$$P(f(x) > f_n^+|X, f) = 1 - \int_{-\infty}^{f_n^+\sigma_n(x)+\mu_n(x)} \frac{1}{\sqrt{2\pi}} exp\left(-\frac{s^2}{2}\right) dt$$

$$P(f(x) > f_n^+|X, f) = 1 - \Phi(\frac{f_n^+ - \mu_n(x)}{\sigma_n(x)})$$

2 Q2

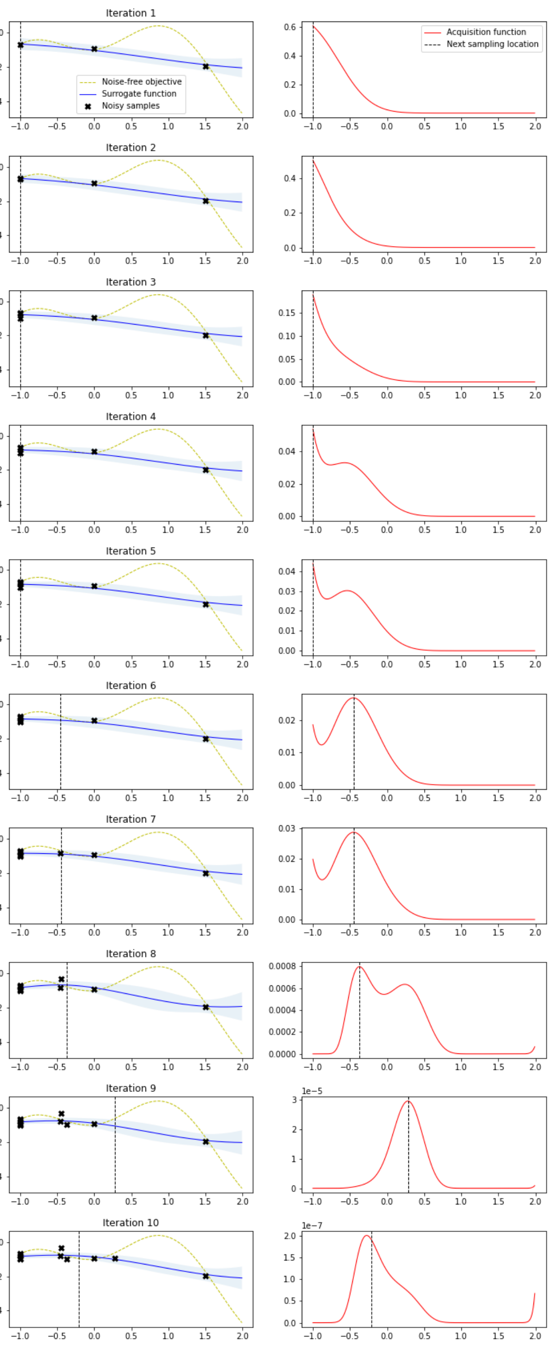
The following code s highly unstable but generates good result a tenth of the time. Therefore we have uploaded the most common version of the process which is not satisfying:

```
'''Calculate expected improvement'''
mu, sigma = gpr.predict(X, return_std=True)

sigma.reshape(-1, 1)
mu.reshape(-1, 1)

fn = np.max(Y_sample)

pi = 1 - norm.cdf((fn - mu)/sigma)
return pi
```

This is the most commonly observed results, the process gets stuck in the corner for the first iteration:

Machine Learning
Seoul National University

Homework #(**3**)
JARRY Guillaume



Machine Learning
Seoul National University

Homework #(**3**)
JARRY Guillaume

---

3 Q2

The probability of improvement acqusition function doesn't seem to work in this case because it is trapped in a local maximum of probability which is clearly not the maximum of the true function. I think this happens because the gradient descent gets stuck in a zone where the gradient is very close to zero, and thus takes time to get out. (Only a very lucky initialization of the function "minimize" in "propose location" produces good result). Therefore, we can help it get unstuck by adding noise to the function, which should prevent the gradient from disappearing. Let's explore the results for a gaussian noise and a uniform noise.

Finally, we can also improve the result of this method by employing an equivalent function whose maximum will be the same as the probability improvement but whose gradient will not vanish.

3.1 Gaussian Noise

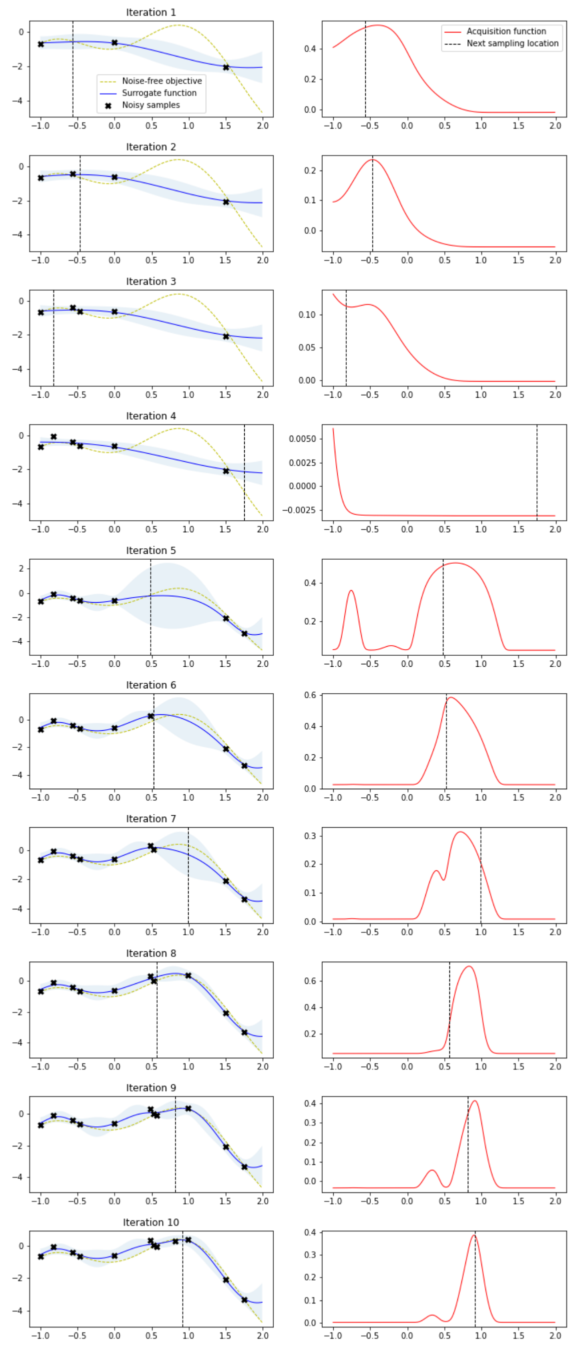
Let's see what the results are by adding gaussian noise to the function according to the following code. (Adding the noise where we added it works here because the gradient is computed by the optimizer using backpropagation, thus adding the noise has an effect on the gradient).

```
'''Calculate expected improvement'''
mu, sigma = gpr.predict(X, return_std=True)

sigma.reshape(-1, 1)
mu.reshape(-1, 1)

fn = np.max(Y_sample)
Xi = np.random.normal(0, )

pi = 1 - norm.cdf((fn - mu)/sigma) + Xi
return pi
```

The optimal value of X_i we found was 0.02.

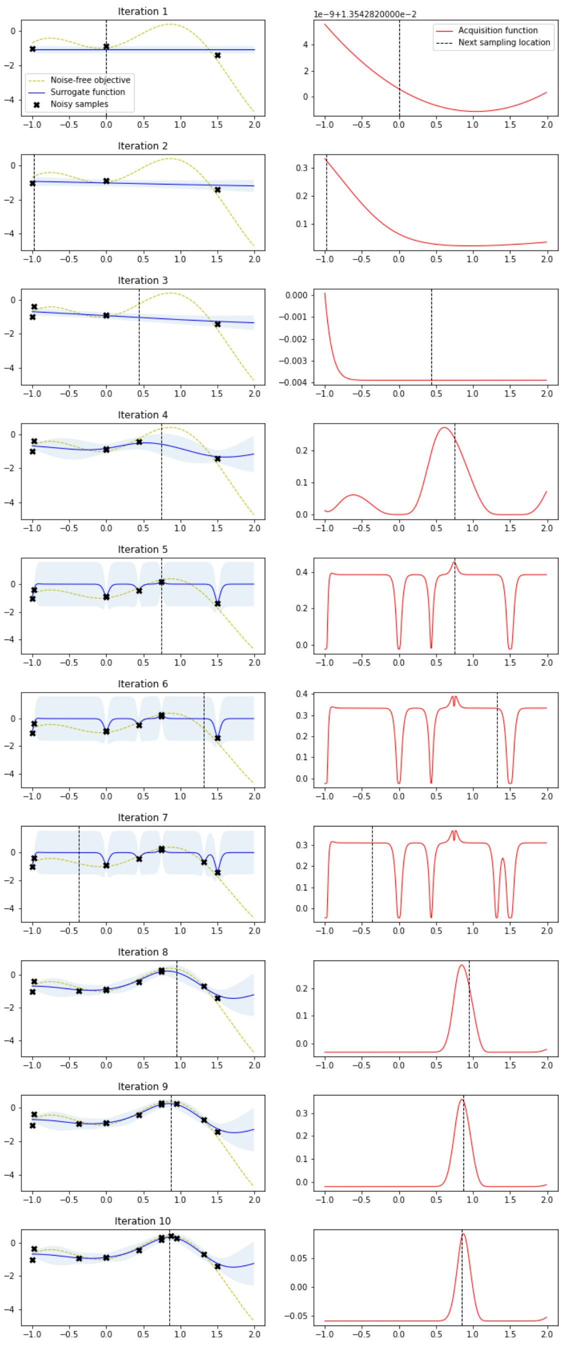
Machine Learning
Seoul National University

Homework #(**3**)
JARRY Guillaume



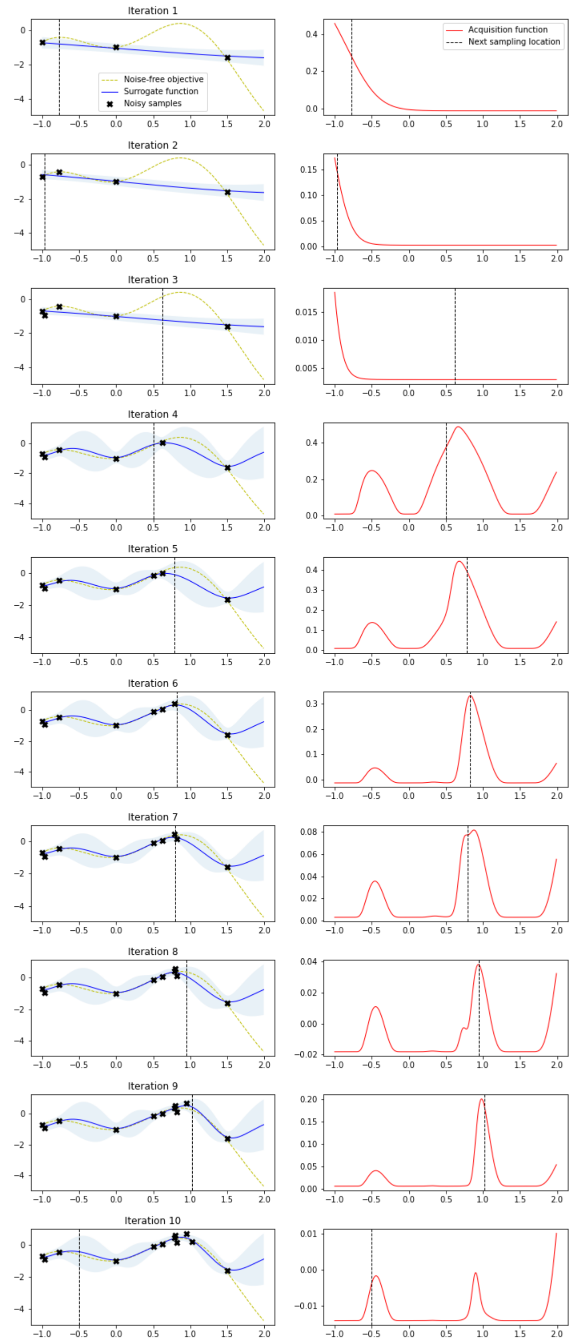
Machine Learning
Seoul National University

Homework #(**3**)
JARRY Guillaume



3.2 Uniform Noise

Maybe a uniform noise might lead to better results. Let's check this out. The process being analogous to a gaussian, we observe roughly the same result.



The above plots were generated by the code:

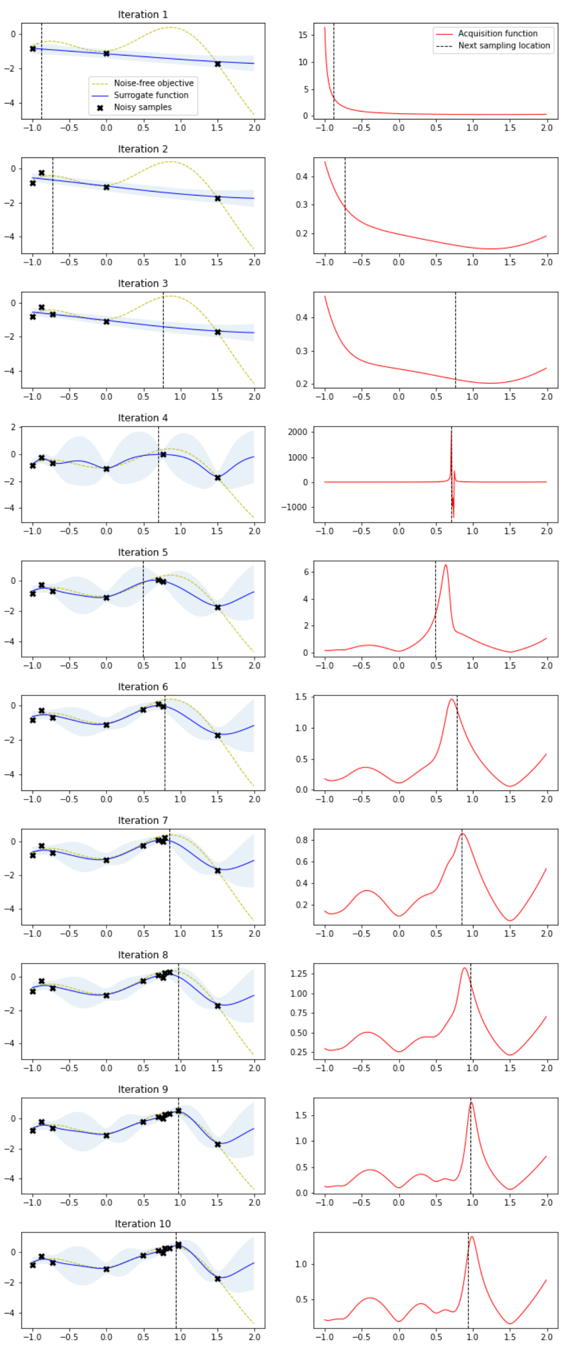
```
'''Calculate expected improvement'''
mu, sigma = gpr.predict(X, return_std=True)
```

```
sigma.reshape(-1, 1)
mu.reshape(-1, 1)

fn = np.max(Y_sample)
Xi = np.random.uniform(-xi, xi)

pi = 1 - norm.cdf((fn - mu)/sigma) + Xi
return pi
```

3.3 Equivalent function + noisy gradient

Another approach consists in noticing that by finding an equivalent function, we can probably escape the problem of the disappearing gradient. Furthermore, the cumulative density function of a Gaussian being strictly increasing, the maximum f_n^+ of the probability improvement will be thef_n that minimizes the expression $\frac{f_n^+ - \mu_n(x)}{\sigma_n(x)}$. Inversing this function ($\frac{\sigma_n(x)}{f_n^+ - \mu_n(x)}$) will not change anything as these function are equivalent in the sens of their minimum. The best results are obtained by combining the noisy gradient descent and the equivalent function. Thest best results are obtained for a Gaussian noise of 0.7.

```
'''Calculate expected improvement'''
mu, sigma = gpr.predict(X, return_std=True)

sigma.reshape(-1, 1)
mu.reshape(-1, 1)

fn = np.max(Y_sample)
Xi = np.random.normal(0, xi)

pi = sigma/(fn - mu) + Xi
return pi
```


Iteration 1
Noise-free objective
Surrogate function
Noisy samples
Acquisition function
Next sampling location
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Iteration 7
Iteration 8
Iteration 9
Iteration 10