1. 이론

1 베이지안 최적화 개요

블랙박스 최적화 문제

블랙박스 최적화 문제는 목적 함수, 제약식, 결정 변수 간 자세한 관계를 알 수 없는 최적화 문제를 말합니다. 구체적으로 추가적인 실험 및 측정 없이 결정 변수에 따른 목적 함숫값을 모르거나 제약식을 만족하는지 알 수 없는 문제를 의미합니다.

일반 최적화 문제

$$\text{maximize } x_1 + x_2$$

$$\text{subject to } x_1^2 + x_2^2 \leq 10$$

- 결정 변수와 목적 함수, 결정 변수와 제약식 간의 관계를 알 수 있는 최적화 문제임
- x_1과 x_2가 각각 2와 1이라면 목적 함수가 3이고 제약식은 $5 \leq 10$으로 만족함을 알 수 있음

블랙박스 최적화 문제

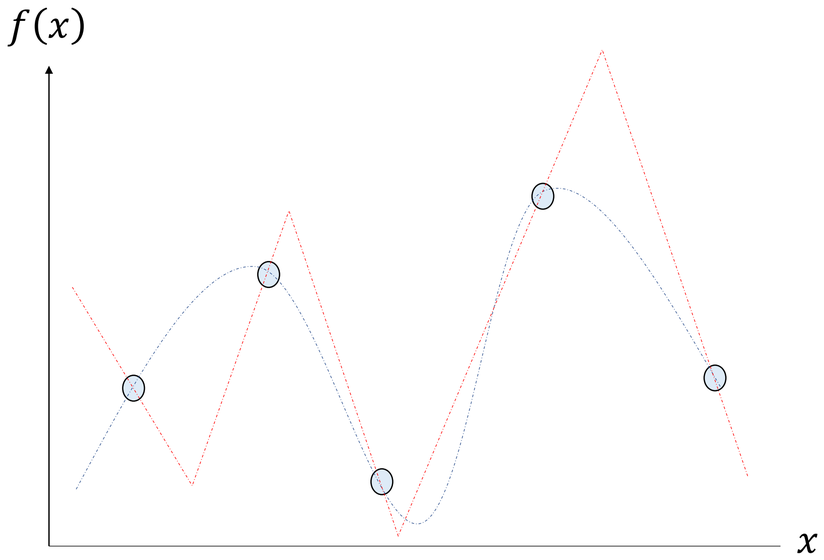
$$\text{maximize } P(\boldsymbol{\lambda}; D)$$

$$\text{subject to } \boldsymbol{\lambda} \in \Lambda$$

- 하이퍼파라미터 튜닝 문제로 블랙박스 최적화 문제임
- 목적 함수에서 $P(\boldsymbol{\lambda}; D)$는 하이퍼파라미터가 $\boldsymbol{\lambda}$인 모델을 데이터 D로 학습했을 때의 성능을 나타냄
- 임의의 해 $\hat{\boldsymbol{\lambda}}$가 제약을 만족하는지는 쉽게 알 수 있지만, 목적 함수는 모델을 학습하고 평가하지 않으면 그 값을 알 수 없음
- 이 문제처럼 블랙박스 최적화 문제는 제약식을 만족하는지는 쉽게 알 수 있지만, 목적 함수를 알 수 없는 경우가 대부분임

블랙박스 최적화 문제의 해법

블랙박스 최적화 문제는 목적 함수를 정확히 알 수 없으므로 경험적으로 해결해야 합니다. 여기서 경험적으로 해결한다는 말은 데이터를 바탕으로 목적 함수를 추론해가면서 탐색 공간을 결정한다는 의미입니다.



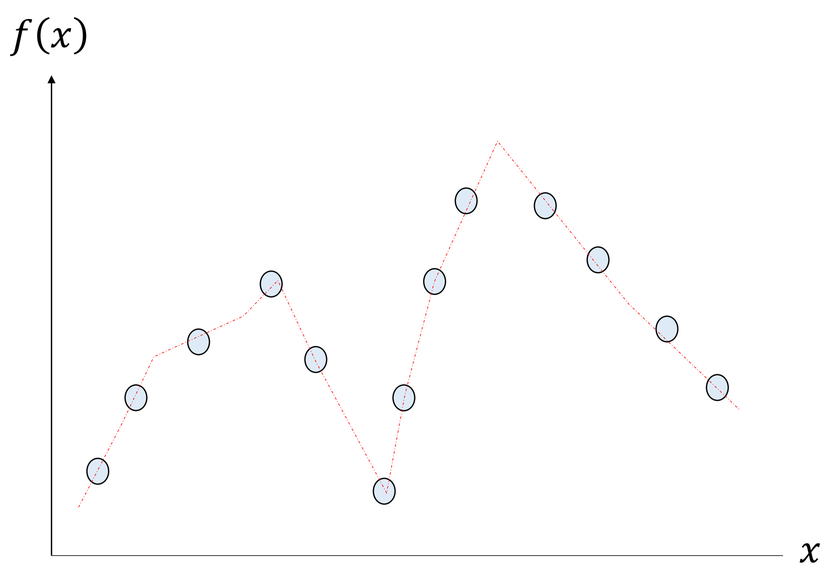
- x와 $f(x)$는 각각 결정 변수와 목적 함수를 나타냄
- 다섯 개의 파란색 원은 실험 등을 통해 측정한 $(x, f(x))$ 꼴의 데이터 포인트를 나타냄
- 다섯 개의 데이터 포인트로 추정할 수 있는 함수는 무한히 많지만, 이 그림에서는 파란색과 빨간색으로 표시된 두 개의 함수만 나타냄
- 두 함수는 그래프 구조가 유사해 보이지만, **최적해의 위치를 다르게 판단함**. 최대화 문제라면 목적 함수를 빨간 함수로 추정했을 때보다 파란 함수로 추정했을 때의 최적해가 더 왼쪽에 있다고 판단함
- 그러나 **데이터 포인트로 함수를 추정한 것**이기에 빨간 함수를 이용한 것과 파란 함수를 이용한 것 중 어느 것이 더 적절한지는 알 수 없음

블랙박스 최적화 문제: 데이터 획득의 어려움

데이터 포인트가 많으면 많을수록 정확한 함수 추정이 가능합니다. 그러나 블랙박스 최적화 문제의 목적 함수를 평가하는 데 시간이 오래 걸려 많은 데이터 포인트를 얻기가 어렵습니다.

많은 데이터 → 정확한 함수 추정 가능

데이터 확보의 어려움



- 예를 들어, 하이퍼파라미터 튜닝 문제라면 특정 하이퍼파라미터를 갖는 모델을 학습하고 평가해야만 하나의 데이터 포인트를 얻을 수 있음
- 심지어는 문제에 따라 금전적인 비용이 발생하기도 함. 예를 들어, 원료 배합을 최적화하는 문제라면 실제로 원료 배합을 평가해야 데이터 포인트를 얻을 수 있는데, 그 과정에서 금전적인 비용이 발생할 수 있음

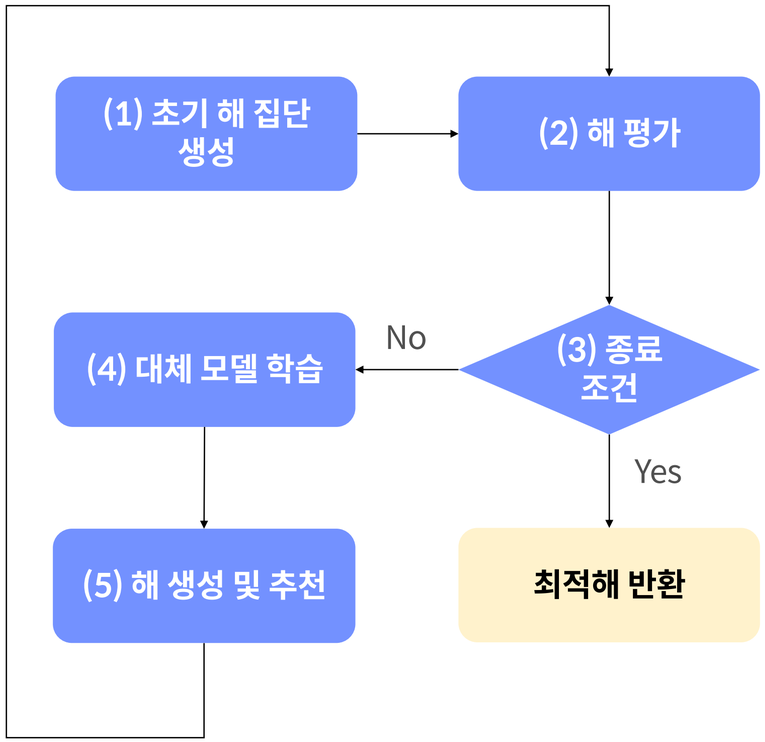
베이지안 최적화가 다루는 문제

베이지안 최적화는 블랙박스 최적화 문제에 대한 대표적인 해법으로 목적 함수가 다음과 같은 특성을 가질 때 적합합니다.

- 함숫값이 특정 구간에서 연속적임
- 목적 함수를 평가하는 데 많은 비용이 발생함
- 도함수를 알 수 없어 내리막 경사법 등을 사용할 수 없음
- 해를 평가하는 과정에서 노이즈(noise)가 발생하지 않는다고 가정할 수 있음

베이지안 최적화의 구성

베이지안 최적화는 샘플러(sampler), 대체 모델(surrogate model), 획득 함수(acquisition function)가 상호 작용하면서 해를 탐색합니다.



(1) 샘플러를 이용해 초기 해 집단 $\mathrm{X}^0 = \{\boldsymbol{x}_1^0, \boldsymbol{x}_2^0, \cdots, \boldsymbol{x}_n^0\}$를 생성합니다. 여기서 X^0은 0번째 이터레이션에서의 해 집단을, $\boldsymbol{x}_i^0$은 0번째로 만든 해 집단의 i번째 해를 나타냅니다.

(2) 목적 함수를 이용하여 해를 평가합니다.

(3) 종료 조건을 만족하는지 확인합니다. 만족한다면 현재까지 찾은 해 가운데 가장 좋은 해를 반환하고 알고리즘을 종료하며, 그렇지 않으면 (4)로 갑니다.

(4) 현재까지 탐색한 해를 사용해 대체 모델을 학습하거나 업데이트합니다. 대체 모델은 특징이 해이고 라벨이 목적 함숫값인 지도 학습 모델입니다.

(5) 샘플러를 사용해 여러 개의 해를 임의로 생성합니다. 생성한 해를 대체 모델과 획득 함수로 평가하여 우수한 해를 추천한 뒤 (2)로 되돌아갑니다.

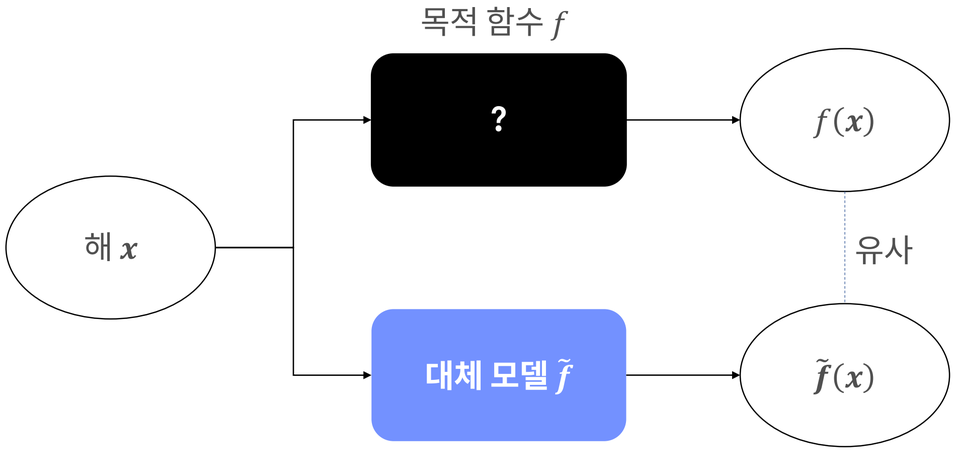
1. 이론

2 대체 모델

대체 모델이란?

대체 모델은 목적 함수를 모사하는 회귀 모델입니다.

대체 모델



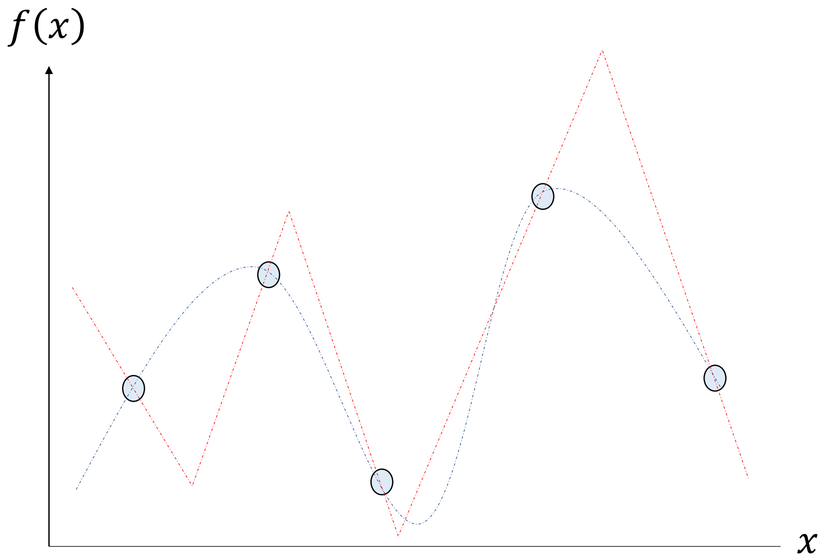
- 대체 모델이 필요한 이유는 목적 함수를 이용해 해를 평가하는 데 많은 시간과 비용이 들기 때문
- 모든 해를 목적 함수를 사용하여 평가하지 않고 대체 모델로 평가한 다음에 평가할 만한 가치가 있는 해만 선별하여 목적 함수를 사용하여 평가함
- 대체 모델은 지도 학습 모델이므로 학습만 되면 빠르게 해를 평가할 수 있음
- 그러나 모든 지도 학습 모델이 그렇듯이 예측 오차가 없을 수는 없어, 열등한 해를 좋다고 평가하는 위험이 있음

대체 모델은 목적 함수를 사용해 해를 평가하는 시간을 단축하는 데 필요함

가우시안 프로세스

가우시안 프로세스(Gaussian Process)는 대체 모델로 가장 많이 사용되는 지도 학습 모델입니다. 그 이유는 가우시안 프로세스가 목적 함숫값의 분포를 추정하는 커널 기반의 비모수 모델 (non-parametric model) [1] 이기 때문입니다.

가우시안 프로세스를 많이 사용하는 이유



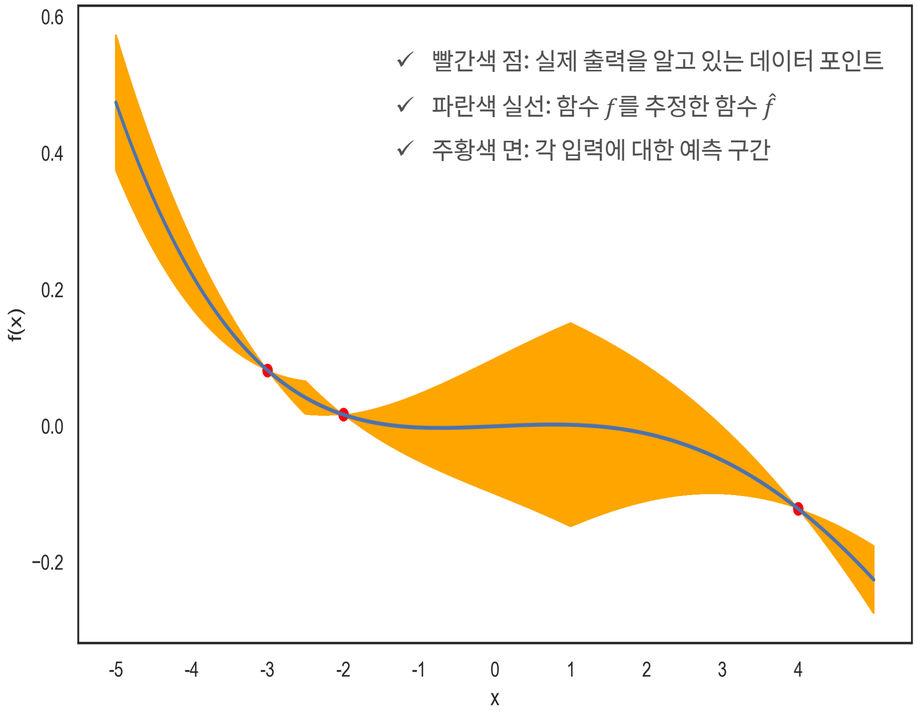
- 다섯 개 데이터 포인트로 추정할 수 있는 함수의 개수는 무한함
- 가우시안 프로세스는 다른 모델처럼 데이터 포인트를 가장 잘 설명하는 특정한 함수를 찾는 것이 아니라, 데이터 포인트를 설명할 수 있는 함수의 분포를 추정함
- 따라서 각 해의 함숫값에 관한 신뢰 구간을 계산할 수 있고, 그에 따라 실제 평가가 필요한 불확실한 해를 선별할 수 있음
- 커널 기반의 비모수 모델이라는 것은 새로운 해의 목적 함숫값을 추정하는 데 목적 함숫값을 알고 있는 기존 해를 활용한다는 뜻임
- 실제 목적 함숫값을 아는 해가 늘어날수록 더 자세한 추정을 할 수 있어 매 이터레이션마다 업데이트해야 하는 대체 모델의 특성에 부합함

[1] 파라미터 수가 학습 데이터 크기에 정비례하는 모델입니다. 예를 들어, 대표적인 모수 모델인 회귀 모델은 데이터 개수에 상관없이 파라미터인 계수와 절편의 개수가 일정합니다. 그러나 가우시안 프로세스와 k-최근접 이웃과 같은 비모수 모델은 학습 데이터가 늘어남에 따라 더 정교해지지만 계산해야 하는 부분이 늘어납니다.

가우시안 프로세스
: 함수에 대한 확률 분포

가우시안 프로세스는 미지의 함수 f에 대한 확률 분포라고 할 수 있습니다. 즉, x가 입력되면 $f(x)$를 출력하는 함수가 있는데, $f(x)$가 하나로 정해지는 것이 아니라 어떤 분포를 따르는 것입니다.

함수 f의 분포 (측정 데이터: $\{(-3, f(-3)), (-2, f(-2)), (4, f(4))\}$)



- 출력을 알고 있는 입력 -3, -2, 4에서는 예측 구간의 길이가 0인데, 실제 출력을 알고 있으니 예측 편차가 없음
- 출력을 알고 있는 입력 근처의 값은 예측 구간이 짧고, $x = 1$과 같이 출력을 알고 있는 입력과 멀리 떨어져 있는 값일수록 예측 구간이 매우 김
- 잘 아는 구간일수록 편차가 적어 예측 구간이 짧지만, 잘 모르는 구간일수록 편차가 커서 예측 구간이 김
- 다시 말해, 관측한 데이터 $\mathrm{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \cdots, \boldsymbol{x}^{(n)}\}$가 있을 때, 새로운 샘플 $\boldsymbol{x}^{(n+1)}$의 함숫값 $f(\boldsymbol{x}^{(n+1)})$는 확률 분포이며 이미 관측된 샘플이 $\boldsymbol{x}^{(n+1)}$와 얼마나 가까운지에 따라 $f(\boldsymbol{x}^{(n+1)})$의 예측 구간의 길이가 결정됨

가우시안 프로세스
: 다변량 정규 분포

가우시안 프로세스는 함숫값이 다변량 정규 분포(multivariate normal distribution)를 따른다고 가정합니다. 다변량 정규 분포를 따르는 확률 변수는 단순히 정규 분포를 따르는 개별 확률 변수의 조합이 아니라, 서로의 분포에 영향을 줍니다.

표현

$$(x_1, x_2) \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$$

- $\mathcal{N}$: 다변량 정규 분포
- $\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$: 평균 벡터(mean vector)
- $\Sigma = \begin{pmatrix} \Sigma_{1,1} & \Sigma_{1,2} \\ \Sigma_{2,1} & \Sigma_{2,2} \end{pmatrix}$: 공분산 행렬(covariance matrix)

확률 밀도 함수

$$\frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)}\left[\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2 - 2\rho\left(\frac{x_1-\mu_1}{\sigma_1}\right)\left(\frac{x_2-\mu_2}{\sigma_2}\right) + \left(\frac{x_2-\mu_2}{\sigma_2}\right)^2\right]\right)$$

- ρ: 두 변수 간 피어슨 상관계수(Pearson correlation coefficient)
- x_1과 x_2가 서로의 분포에 영향을 주며, 영향을 주는 정도는 공분산으로 표현됨

다변량 정규 분포에 대해 깊은 이해가 중요한 것이 아니라, 각 변수가 서로 영향을 준다는 것이 중요함

가우시안 프로세스
: 다변량 정규 분포 (계속)

가우시안 프로세스는 학습 데이터 $f(\boldsymbol{x}^{(1:n)}) = \left(f(\boldsymbol{x}^{(1)}), f(\boldsymbol{x}^{(2)}), \cdots, f(\boldsymbol{x}^{(n)})\right)$와 새로 예측할 $f(\boldsymbol{x}^{(n+1)})$ 모두 같은 다변량 정규 분포를 따른다고 간주합니다.

학습 데이터의 분포

$$f(\boldsymbol{x}^{(1:n)}) \sim \mathcal{N}\left(m(\boldsymbol{x}^{(1:n)}), K(\boldsymbol{x}^{(1:n)})\right)$$

- m: 평균 함수(mean function)로 각 변수의 평균을 예측하는 함수
- K: 커널 함수(kernel function)로 변수 간 유사도를 반영하는 함수
- 두 함수 모두 사용자가 정의하지만, 커널 함수가 더 중요한 하이퍼파라미터임

조건부 분포

$$f(\boldsymbol{x}^{(n+1)}) \mid f(\boldsymbol{x}^{(1:n)}) \sim \mathcal{N}(\mu_{n+1}, \sigma_{n+1}^2)$$

- μ_{n+1}: $f(\boldsymbol{x}^{(n+1)})$의 평균, $m(\boldsymbol{x}^{n+1})$
- σ_{n+1}^2: $f(\boldsymbol{x}^{(n+1)})$의 표준편차, $K(\boldsymbol{x}^{1:n}, \boldsymbol{x}^{n+1})$

가우시안 프로세스
: 학습

가우시안 프로세스 모델의 학습은 커널에 포함된 파라미터를 추정하는 것입니다.

커널 함수 예시: RBF 커널

$$\mathrm{K}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp(-\gamma \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2)$$

- γ: RBF 커널의 하이퍼 파라미터

가우시안 프로세스
: 문제점

가우시안 프로세스는 다변량 정규분포를 가정하기 때문에 몇 가지 문제점이 있습니다.

Frazier[1]에 따르면 20차원 이하의 실수 벡터에만 적합함

범주형 변수를 포함하는 모델 선택과 하이퍼파라미터 튜닝 문제에 가우시안 프로세스를 적용하는 것은 부적절합니다.

계층적 구조에 부적합함

예를 들어, 모델 선택과 하이퍼파라미터 튜닝 문제에서 모델 변수가 랜덤 포레스트면 튜닝해야 하는 하이퍼파라미터는 나무의 최대 깊이, 나무 개수 등입니다. 그러나 모델 변수가 신경망이면 튜닝해야 하는 하이퍼파라미터는 은닉층의 구조가 됩니다. 이처럼 한 변수의 값에 따라 다른 변수를 사용하거나 사용하지 못하는 상황에서 가우시안 프로세스를 적용하기는 어렵습니다.

어디까지나 적용했을 때 좋은 성능을 기대하기 어렵다는 이야기지 적용할 수 없다는 것은 아님

실제로 많은 연구에서 가우시안 프로세스를 연속형 변수로만 구성된 저차원의 데이터가 아닌 데이터에도 적용했습니다. 또한, 특수한 상황에 맞게 가우시안 프로세스 모델을 수정 합니다.

[1] Frazier, P. I. (2018). A tutorial on Bayesian optimization. arXiv preprint arXiv:1807.02811

1. 이론

3 샘플러와 대체 모델 실습

예제

샘플러와 대체 모델 실습을 진행할 예제는 다음과 같습니다.

- **목적 함수:** maximize $f(x_1, x_2) = x_1^2 \sin\big((5\pi(-x_1 + 2x_2)\big)$
- **제약식:** $0 \leq x_1, x_2 \leq 1$

- 목적 함수가 복잡하긴 하나, 사실 블랙박스 최적화 문제는 아님

목적 함수 정의

```
1  import numpy as np
2  def obj_func(x):
3          return x[0] ** 2 * np.sin(5 * np.pi * (-x[0] + 2 * x[1]))
```

- **라인 2:** 간결한 코드 작성을 위해 x1과 x2를 따로 입력받지 않고 샘플 x를 입력받습니다. x[0]를 x1으로, x[1]을 x2로 보면 됩니다.

샘플러 정의

x1과 x2가 모두 0과 1 사이의 값이므로 np.random.random 함수를 사용해 샘플러를 정의합니다.

샘플러 정의

```
1  def sampler(n):
2      return np.random.random((n, 2))
```

- **라인 1 - 2:** 샘플 개수 n을 입력받아 (n, 2) 크기의 난수 행렬을 반환합니다.

샘플링

```
1  X = sampler(100)
2  y = np.apply_along_axis(obj_func, axis = 1, arr = X)
```

- **라인 1:** 샘플러를 사용해 100개의 해를 생성해 X에 저장합니다.
- **라인 2:** X에 속한 모든 요소를 obj_func으로 평가합니다.

가우시안 프로세스

사이킷런의 GaussianProcessRegressor 클래스를 사용하면 가우시안 프로세스 모델을 만들 수 있습니다. 이렇게 만든 가우시안 프로세스 모델은 다른 사이킷런의 인스턴스와 마찬가지로 predict 메서드를 이용해 라벨을 예측할 수 있습니다. 차이가 있다면 가우시안 프로세스 모델은 return_std와 return_cov라는 인자로 표준편차와 다른 샘플과의 공분산을 출력할 수 있다는 점입니다.

주요 인자

인자	설명
kernel	가우시안 프로세스의 커널 함수를 지정합니다. sklearn.gaussian_process.kernels에 있는 커널 함수를 조합하여 설정할 수 있습니다.

가우시안 프로세스 모델 학습 예제

```
1  from sklearn.gaussian_process import GaussianProcessRegressor as GPR
2  from sklearn.gaussian_process.kernels import RBF, WhiteKernel
3  model = GPR( kernel=RBF() + WhiteKernel(), random_state=2022).fit(X, y)
```

- **라인 3:** RBF 커널과 노이즈를 나타내는 WhiteKernel을 더한 커널을 갖는 가우시안 프로세스 모델을 X와 y로 학습합니다.

1. 이론

4 획득 함수

획득 함수란?

획득 함수는 실제 출력을 모르는 해집합 X_{new}에 속한 모든 해를 평가하고 그 가운데 가장 값이 큰 해를 추천합니다. 상황에 따라서는 상위 k개의 해를 추천하기도 합니다.

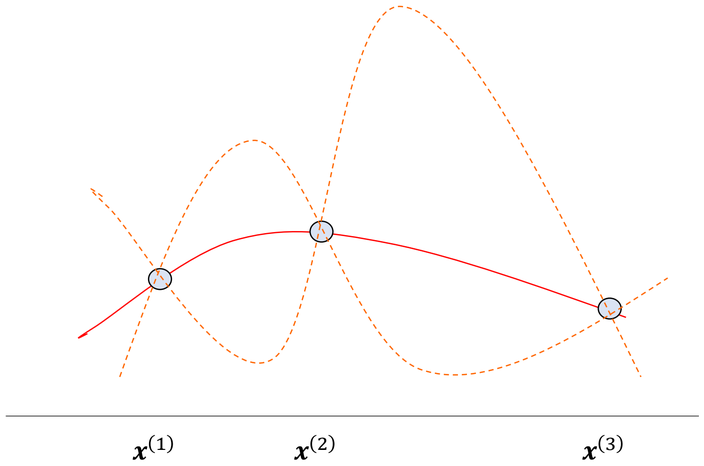
$$\boldsymbol{x}_{new}^{*} = \underset{\boldsymbol{x} \in X_{\text{new}}}{\operatorname{argmax}}\, g(\boldsymbol{x})$$

- g: 획득 함수
- 추천된 해 $\boldsymbol{x}_{\text{new}}^{*}$는 실제로 평가를 진행하여 $\left(\boldsymbol{x}_{\text{new}}^{*}, f(\boldsymbol{x}_{\text{new}}^{*})\right)$를 포함하여 가우시안 프로세스 모델을 재학습함

획득 함수의 전략

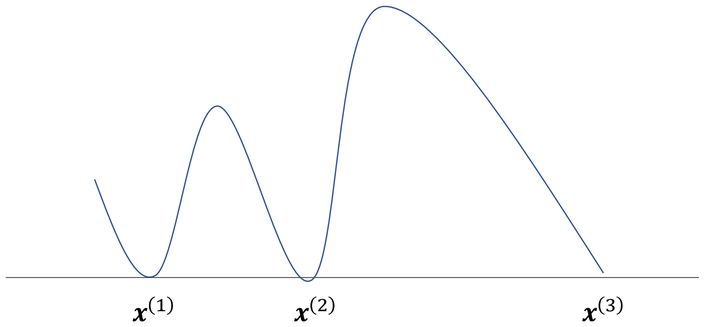
획득 함수는 착취(exploitation)와 탐색(exploration) 전략으로 구성됩니다. 착취 전략이란 좋은 해 근처의 해를 찾는 전략을, 탐색 전략이란 아직 탐색되지 않은 구간을 중심으로 찾는 전략을 의미합니다.

목적 함수



- 현재까지 세 개의 입력 $x^{(1)}, x^{(2)}, x^{(3)}$을 평가했음
- 빨간 실선: 추정한 목적 함수의 평균
- 주황색 점선: 추정한 목적 함수의 예측 구간

획득 함수



- $f(x^{(1)}), f(x^{(2)}), f(x^{(3)})$에 대한 편차는 0임
- $x^{(1)}, x^{(2)}, x^{(3)}$에 대한 출력을 알고 있으므로, 평가할 가치가 없어 획득 함숫값이 0임
- 편차가 크고 상대적으로 값도 크리라 예상되는 $x^{(2)}$와 $x^{(3)}$사이에 있는 값의 획득 함숫값이 다른 구간에 있는 값에 대한 획득 함숫값보다 큽니다.

개선 확률 함수

개선 확률 함수는 현재까지 찾은 최대 출력값보다 더 큰 함숫값을 도출할 확률입니다.

일반식

$$\mathrm{PI}(\boldsymbol{x}^{new}) = \Pr\left(f(\boldsymbol{x}^{new}) > \max_i f\left(\boldsymbol{x}^{(i)}\right)\right)$$

- $\mathrm{PI}(\boldsymbol{x}^{new})$: $\boldsymbol{x}^{new}$의 개선 확률

가우시안 프로세스에서의 계산식

$$\mathrm{PI}(\boldsymbol{x}^{new}) = \Phi\left(\frac{\mu_{new} - \mu_{best}}{\sigma_{new}}\right)$$

- μ_{new}: 가우시안 프로세스가 예측한 $\boldsymbol{x}^{new}$의 함숫값의 평균
- σ_{new}: 가우시안 프로세스가 예측한 $\boldsymbol{x}^{new}$의 함숫값의 표준편차
- μ_{best}: 현재까지 탐색한 모든 함숫값의 최대 평균
- Φ: 누적 분포 함수(cumulative distribution function)

개선 확률 함수 실습

파이썬으로 개선 확률 함수를 구현해보겠습니다.

파이썬에서의 개선 확률 함수 예시

```
1  from scipy.stats import norm
2  def POI(X_new, model, best_mu):
3      mu, sigma = model.predict(X_new, return_std = True)
4      output = norm.cdf((mu-best_mu)/(sigma + 0.0000000001))
5      return output
```

- **라인 2:** 새로운 입력 X_new와 가우시안 프로세스 모델인 model, 현재까지 가장 큰 평균인 best_mu를 입력으로 하는 개선 확률 함수 POI를 정의합니다.
- **라인 3:** return_std 인자를 True로 설정해서 새로운 입력에 대한 평균과 표준편차를 계산하고 그 값을 mu와 sigma에 저장합니다.
- **라인 4:** 표준 정규 분포를 따르는 변수에 대해 (mu-best_mu)/(sigma + 0.0000000001)보다 작을 확률을 output에 저장합니다. 이때, 분모가 0이 될 수 있으므로 분모에 매우 작은 양수를 더 했습니다.

개선 확률 함수를 이용한 해 추천 예시

```
1  best_mu = max(model.predict(X))
2  X_new = sampler(100)
3  score_list = POI(X_new, model, best_mu)
4  print(score_list.argmax())
```

- **라인 1:** 현재까지 탐색한 모든 해집합인 X의 평균값을 계산하고 그 가운데 가장 큰 값을 best_mu에 저장합니다.
- **라인 2:** 새로운 100개의 해를 생성하여 X_new에 저장합니다.
- **라인 3:** POI 함수를 사용해 X_new에 속한 각 해의 점수를 계산합니다.
- **라인 4:** argmax 메서드를 이용해 score_list에서 값이 가장 큰 인덱스를 출력합니다.

```
74
```

개선 기대 함수

가장 널리 쓰이는 획득 함수인 개선 기대 함수는 새로운 해를 탐색함으로써 기대되는 함숫값의 갱신 정도를 나타냅니다.

일반식

$$\mathrm{EI}(\boldsymbol{x}^{new}) = \mathbb{E}\left(\max\left(f(\boldsymbol{x}^{new}) - \max_i\left(f\left(\boldsymbol{x}^{(i)}\right)\right), 0\right)\right)$$

- $\mathbb{E}$: 기댓값(expected value)
- 새로운 해 $f(\boldsymbol{x}^{new})$가 지금까지 탐색했던 가장 좋은 해 $\max_i\left(f\left(\boldsymbol{x}^{(i)}\right)\right)$보다 좋다면 기댓값이 양수가 되고, 그렇지 않으면 개선되는 것이 없으므로 0이 됨

개선 기대 함수 (계속)

가장 널리 쓰이는 획득 함수인 개선 기대 함수는 새로운 해를 탐색함으로써 기대되는 함숫값의 갱신 정도를 나타냅니다.

가우시안 프로세스에서의 계산식

$$\mathrm{EI}(\boldsymbol{x}^{new}) = \begin{cases} \left(\mu_{new} - \max_i \left(f\left(\boldsymbol{x}^{(i)}\right)\right) - \varepsilon\right)\Phi(Z) + \sigma_{new}\phi(Z) & \text{, if } \sigma_{new} > 0 \\ 0 & \text{, if } \sigma_{new} = 0 \end{cases}$$

- ε: σ에 대한 가중치
- ϕ: 확률 밀도 함수

$$Z = \begin{cases} \dfrac{\left(\mu_{new} - \max_i \left(f\left(\boldsymbol{x}^{(i)}\right)\right) - \varepsilon\right)}{\sigma_{new}} & , if\ \sigma_{new} > 0 \\ 0 & , if\ \sigma_{new} = 0 \end{cases}$$

개선 기대 함수 실습

파이썬으로 개선 기대 함수를 구현해보겠습니다.

파이썬에서의 개선 기대 함수 예시

```
1  def EI(X_new, model, best_mu, e = 0.01):
2      mu, sigma = model.predict(X_new, return_std = True)
3      z = np.zeros(len(X_new))
4      z[sigma > 0] = ((mu - best_mu - e) / sigma)[sigma > 0]
5      return (mu - best_mu - e) * norm.cdf(z) + sigma * norm.pdf(z)
```

- **라인 2**: 새로 입력된 X_new에 대해 예측한 평균과 표준편차를 각각 mu와 sigma에 저장합니다.
- **라인 3**: z를 영벡터로 초기화합니다.
- **라인 4**: sigma가 0인 z는 그대로 0으로, 그렇지 않으면 z를 계산합니다. **이전 식**과 대조하자면, z는 Z를, best_mu는 $\max_i \left(f\left(\boldsymbol{x}^{(i)}\right)\right)$를, e는 ε을, sigma는 σ_{new}를 나타냅니다.
- **라인 5**: 개선 기댓값을 계산하여 반환합니다.

$$\mathrm{EI}(\boldsymbol{x}^{new}) = \begin{cases} \left(\mu_{new} - \max_i \left(f\left(\boldsymbol{x}^{(i)}\right)\right) - \varepsilon\right)\Phi(Z) + \sigma_{new}\phi(Z) & \text{, if } \sigma_{new} > 0 \\ 0 & \text{, if } \sigma_{new} = 0 \end{cases}$$

$$Z = \begin{cases} \dfrac{\left(\mu_{new} - \max_i \left(f\left(\boldsymbol{x}^{(i)}\right)\right) - \varepsilon\right)}{\sigma_{new}} & , if\ \sigma_{new} > 0 \\ 0 & , if\ \sigma_{new} = 0 \end{cases}$$

개선 기대 함수를 이용한 해 추천 예시

```
1  best_mu = max(model.predict(X))
2  X_new = sampler(100)
3  score_list = EI(X_new, model, best_mu)
4  print(score_list.argmax())
```

```
82
```

1. 이론

5 메인 코드

메인 코드

획득 함수는 개선 기대 함수를 사용하고 각 이터레이션마다 100개의 해를 샘플링한 뒤 획득 함숫값이 가장 큰 하나의 해만 추가하는 이터레이션을 100번 반복하는 메인 함수를 작성하겠습니다.

베이지안 최적화의 메인 함수 예시

```
1   def main(n, num_iter):
2       X = sampler(n)
3       y = np.apply_along_axis(obj_func, axis = 1, arr = X)
4
5       for _ in range(num_iter):
6           model = GPR(kernel=RBF() + WhiteKernel(), random_state=2022).fit(X, y)
7           best_mu = max(model.predict(X))
8           X_new = sampler(n)
9           score_list = EI(X_new, model, best_mu)
10          x_new = X_new[score_list.argmax()]
11          y_new = obj_func(x_new)
12          X = np.vstack([X, x_new])
13          y = np.append(y, y_new)
14
15      return X[y.argmax()], y.max()
```

- **라인 2~3**: n 개의 샘플을 임의로 샘플링하여 X에 저장하고 X에 있는 각 샘플을 평가한 결과를 y에 저장합니다.
- **라인 6**: 현재까지 탐색한 X와 y를 사용해 가우시안 프로세스 모델을 학습합니다.
- **라인 7**: 현재까지 탐색한 X에 대한 평균의 최댓값을 best_mu에 저장합니다. 여기서 X를 입력으로 하지 않고 X_new만 입력하여 최댓값을 갱신할 수도 있습니다. 그러나 예측 자체에는 오랜 시간이 걸리지 않으므로 편의상 중복이 발생하더라도 X를 입력으로 사용했습니다.
- **라인 8**: 새로운 n 개의 값을 추가로 샘플링합니다.
- **라인 9**: EI 함수를 사용해 X_new에 있는 샘플을 평가하고 그 결과를 score_list에 저장합니다.
- **라인 10~11**: EI가 가장 높은 입력을 x_new에 저장하고 x_new를 평가한 결과를 y_new에 저장합니다.
- **라인 12~13**: x_new와 y_new를 각각 X와 y에 추가합니다.

메인 코드 (계속)

획득 함수는 개선 기대 함수를 사용하고 각 이터레이션마다 100개의 해를 샘플링한 뒤 획득 함숫값이 가장 큰 하나의 해만 추가하는 이터레이션을 100번 반복하는 메인 함수를 작성하겠습니다.

베이지안 최적화의 메인 함수 예시

```
1  print(main(100, 100))
```

```
(array([0.99161018,       0.7484802 ]),     0.9798203596022771)
```