Bayesian Optimization

베이즈 스터디
박준민

Optimization

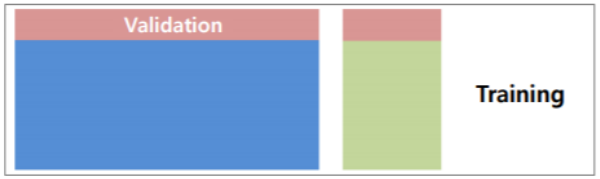
- Lasso Linear Regression 하이퍼파라미터 튜닝해라

$$\hat{y}_i = \widehat{\beta_0} x_{i0} + \widehat{\beta_1} x_{i1} + \widehat{\beta_2} x_{i2} + \ldots + \widehat{\beta_p} x_{ip}$$

$$\min_\beta \left\{ \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{p} x_{ij} \beta_j \right)^2 + \lambda \sum_{j=0}^{p} |\beta_j| \right\}$$

λ : $0.001, 0.01, 0.1, 1.0, 10.0, \ldots$

10 fold cross validation



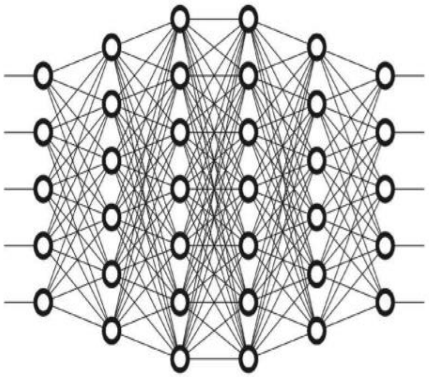
1시간

10개 하이퍼파라미터 후보
× 1 시간 = 10시간

=> 오키 개노가다긴 한데 해줄게

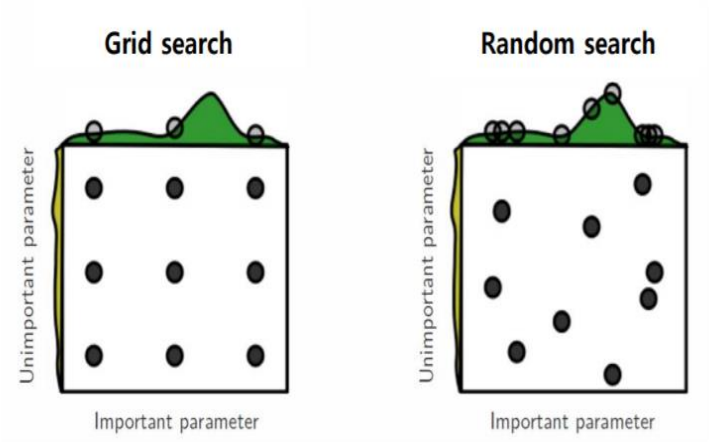
Optimization

- Neural Network 모델 하이퍼파라미터 튜닝해라



Hyperparameters

- Learning rate
- # of iterations
- Minibatch size
- # of hidden layers
- # of hidden nodes
- Type of activation functions
- .....



=> 한 조합에 1시간인데 선넘지

+ 어떤 조합이 잘 나올지 모르고 무식하게 모든 조합 무식하게 다 떼려 박게?

Basic of Bayesian Optimization

- 1970년대 80년대 처음
- $x^* = \arg\min_{x \in X} f(x).$

1. 관측된 data(D = $\{x_1, x_2, \cdots, x_n\}$)를 사용해 f(x)를 **어떤 방식을 통해 estimate**
2. f(x)를 더 정밀히 예측하기 위해 x_{new}를 **어떤 decision rule을 통해 선택**
3. x_{new}를 D에 추가하고 적당한 stopping criteria에 도달할 때까지 1로 돌아가 반복

Basic of Bayesian Optimization

- 1번 단계에서 f(x)가 Gaussian process prior를 가진다고 가정한 다음, posterior를 계산하여 f(x)를 estimate
- 2번 단계에서 acquisition funtion $a(x|D)$를 디자인하고 $\arg\max_x a(x|D)$를 계산하여 x_{new}를 선택

Learning rate	# of iterations	# of hidden layers	# of hidden nodes	...	Mini batch size
0.001	1000	3	16	...	128
0.001	100	3	16	...	128
0.001	1000	5	16	...	128

1 관계 파악 (사전지식)

Generalization performance
100
200
300

2 의사결정

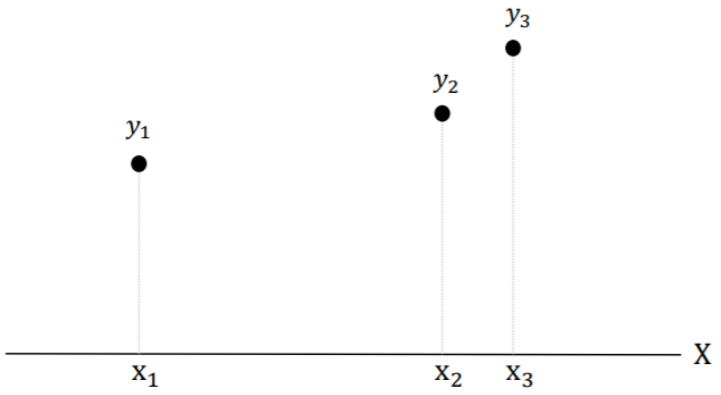
Surrogate Model

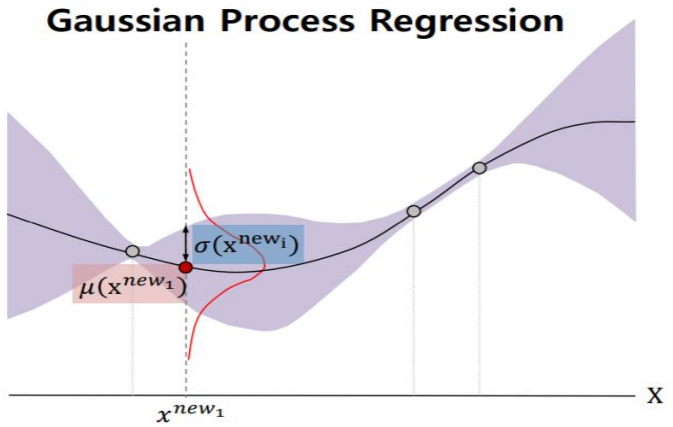
- Gaussian Process Regression으로 f(x) 추정

- Guassian Process란?
  다변량 분포

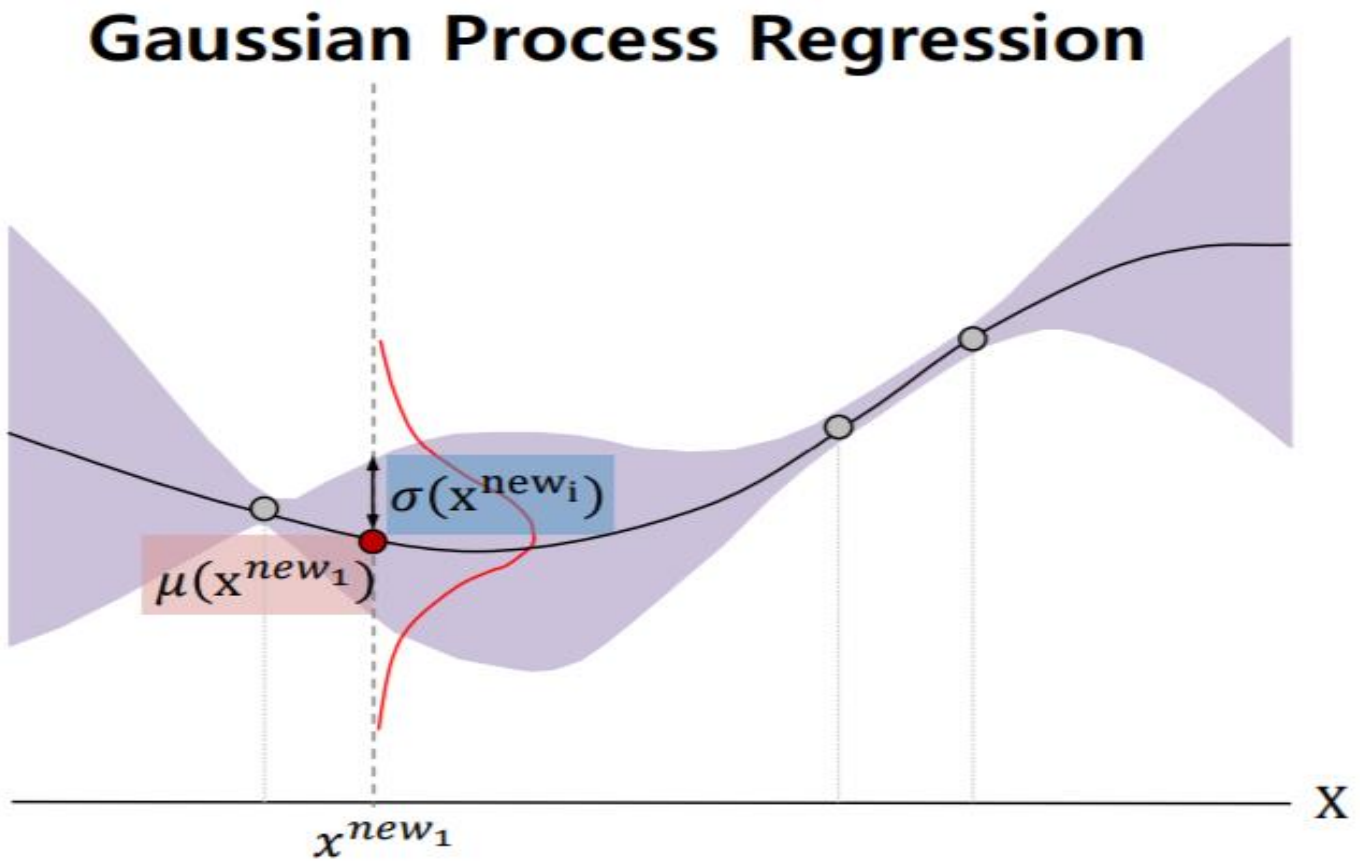
$$p(x, y) \sim \mathcal{N}\left(\begin{bmatrix}\mu_x \\ \mu_y\end{bmatrix}, \begin{bmatrix}\Sigma_x \Sigma_{xy} \\ \Sigma_{xy}^T \Sigma_y\end{bmatrix}\right)$$

  무한 차원으로 확대
  일종의 함수에 대한 분포 $P(X) \sim GP(m(t), k(x, x\prime))$





Surrogate Model



Surrogate Model

- Gaussian Process의 예시

$$P(X) \sim GP(m(t), k(x, x\prime))$$

$$m(x) = 0$$ (주로 constant value or 0)

$$k(x, x\prime) = \theta_1 exp(-\frac{\theta_2}{2}(x - x\prime)^2)$$ (kernel function - Squared Exponential)

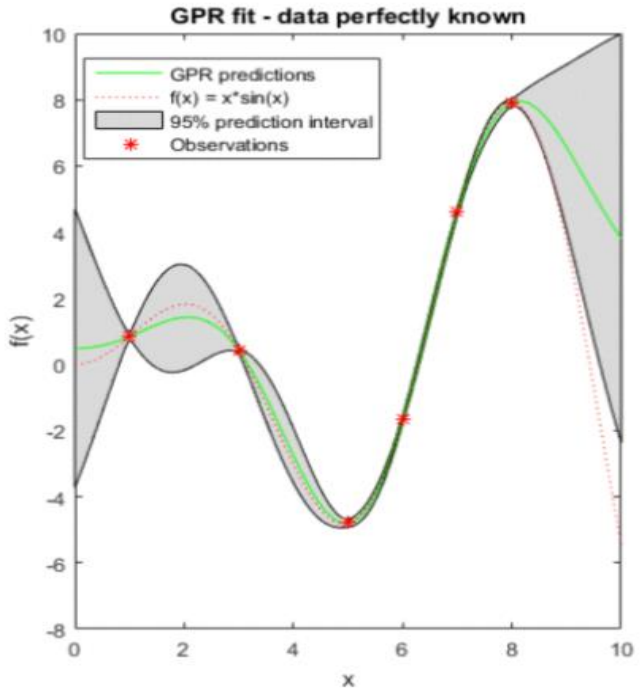
kernel function은 Squared Exponential 외에도
Matern, Periodic 등등 다양하게 설정 가능

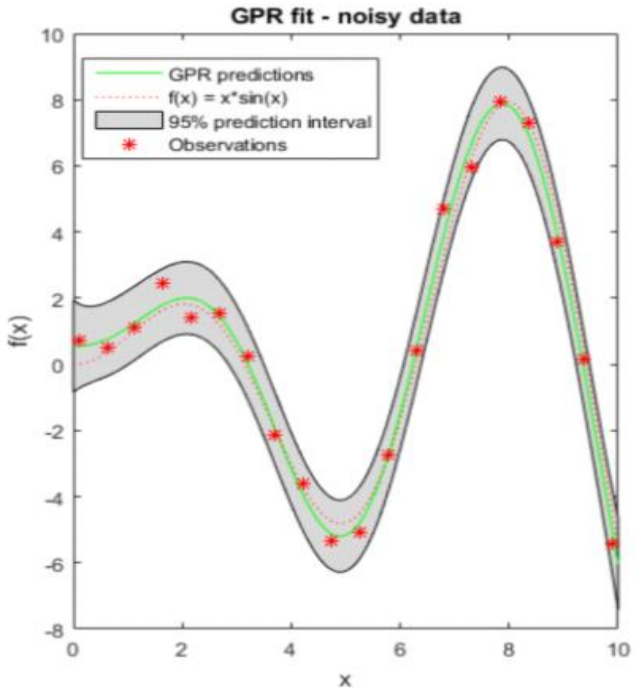
Surrogate Model

- Gaussian Process with Noisy data

$y_i = \mathrm{f}(x_i) + \varepsilon_i \quad (\varepsilon_i \sim N(0, \sigma^2))$

$=> \quad y_i \sim N(\mathrm{f}(x_i), \sigma^2)$





Acquisition Function

- Gaussian Process 가정을 통해 f(x)추정
- 모든 x points에서 mean, variance 계산 가능
- 다음 x_{new}를 어떤 기준으로 정할 것인가?

  1. estimated mean의 값이 가장 작은 지점을 관측하여, 현재까지 관측된 값들을 기준으로 가장 좋은 점 (exploit)
  2. estimated variance의 값이 가장 큰 지점을 관측하여, 함수의 모양을 더 정교하게 탐색 (explore)

- Acquisition function : explore - exploit을 적절하게 균형을 잡아주는 역할

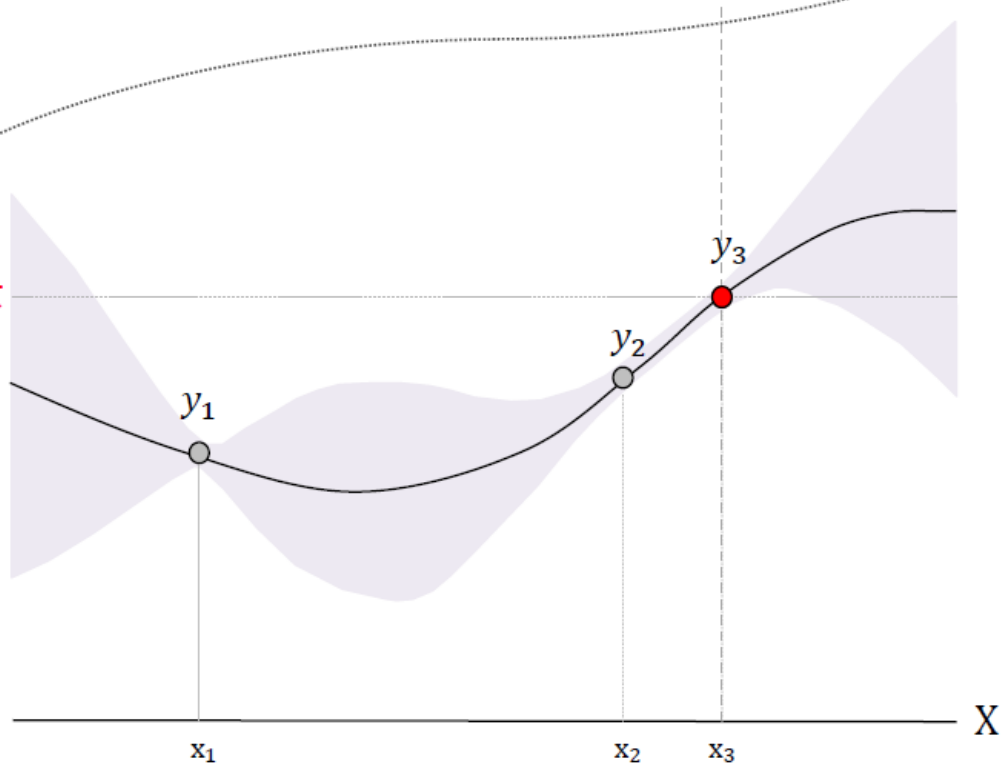
  ex) Probability of Improvement, Expected Improvement, UCB, ….

Acquisition Function

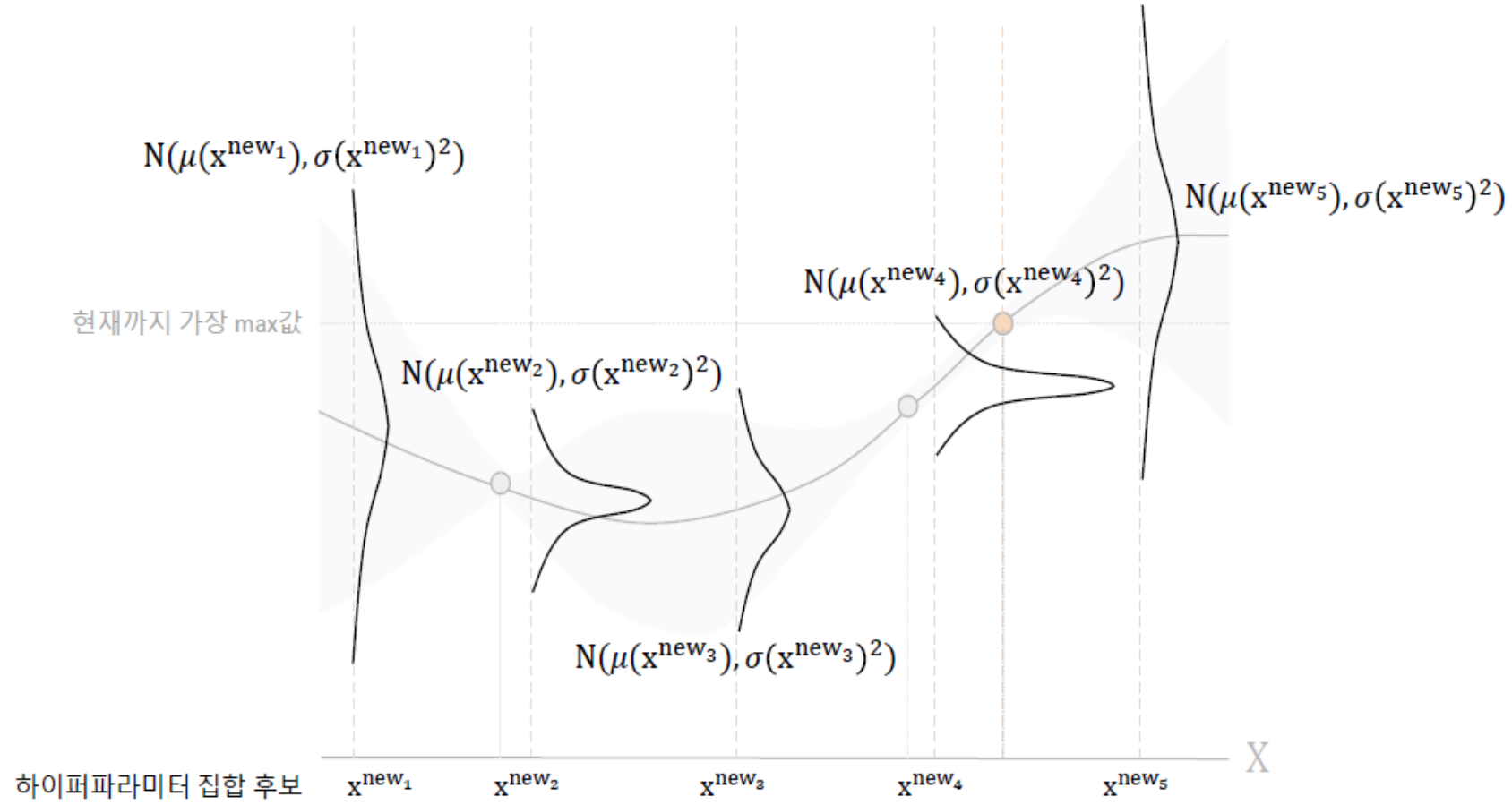
$$x^* = \underset{x^{new_i} \in X^{new}}{\operatorname{argmax}} \; \text{Expected Improvement}(x^{new_i})$$

$$:= E\left(\max\left(0, \left[f(x^{new_i}) - \max_{x_j \in X} f(x_j)\right]\right)\right)$$

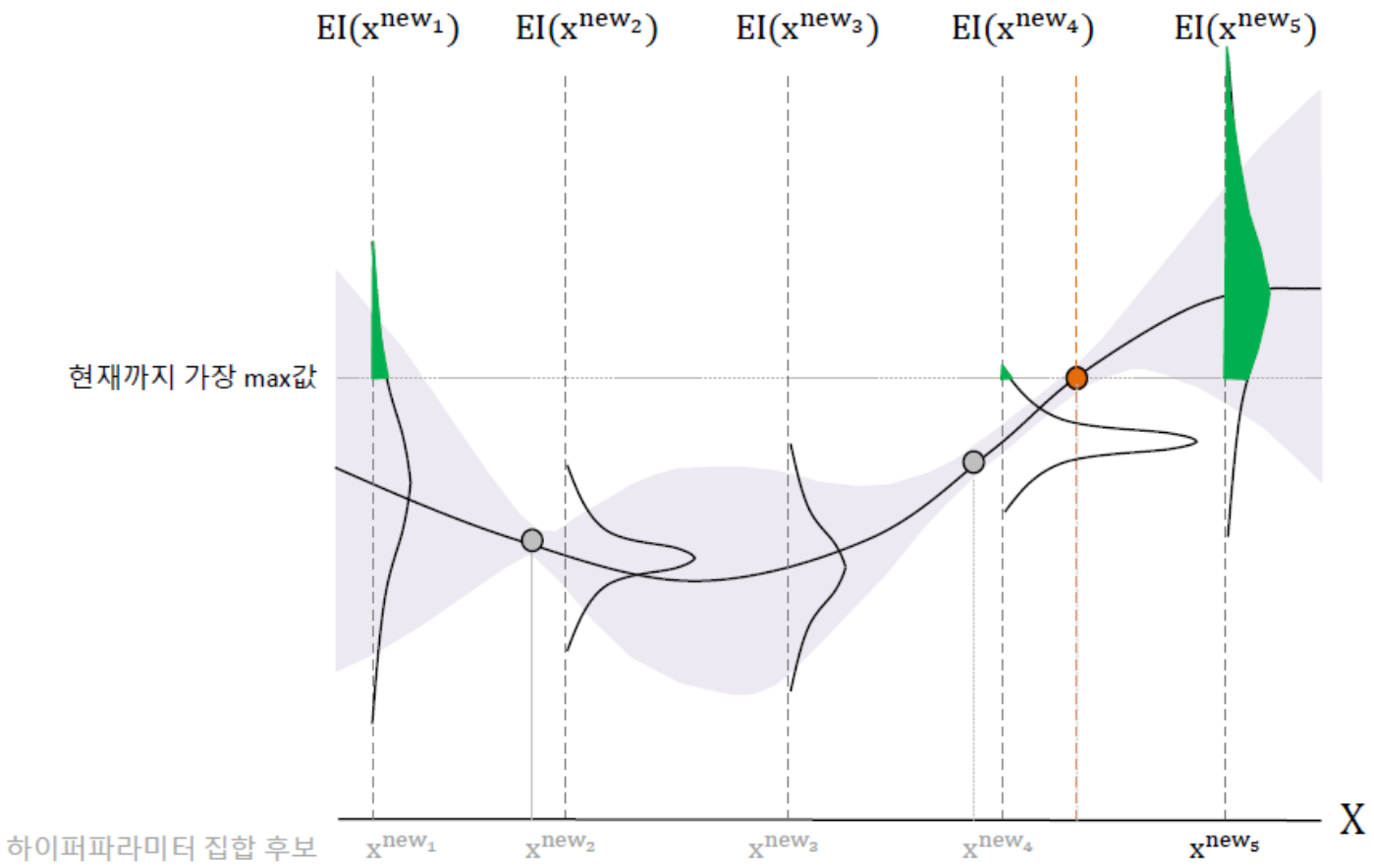
현재까지 가장 max값



Acquisition Function



Acquisition Function



$= x^*$

Limit of Bayesian Optimization

- 상당히 impractical하다

  1. Hyperparameter를 search하기 위해 BO를 사용하는데, BO를 사용하기 위해서는 GP의 hyperparameter들을 튜닝해야한다
     (kernel function의 parameter 등)

  2. 어떤 stochastic assumption을 하느냐에 따라 (어떤 kernel function을 사용해야할지 등) 결과가 천차만별로 바뀌는데, 어떤 선택이 가장 좋은지에 대한 가이드가 전혀 없다
     (model selection에 민감)

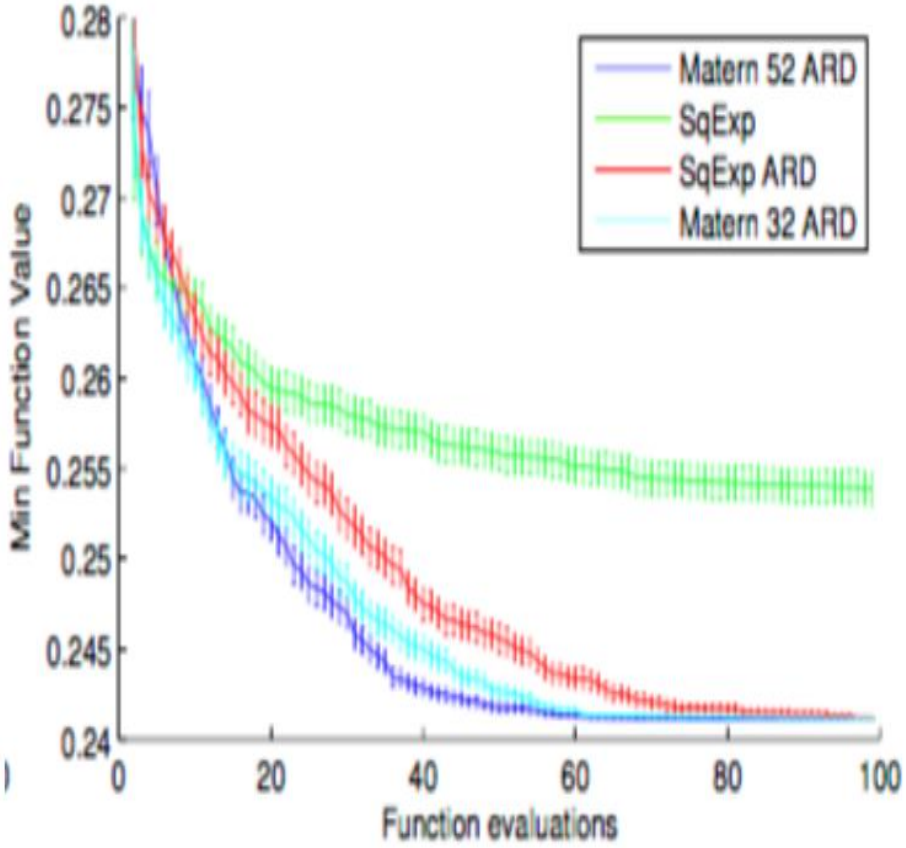
  3. Acquisition function을 사용해 다음 지점을 찾는 과정 자체가 sequential하기 때문에 grid search나 random search와는 다르게 parallelization이 불가능하다.

  4. 위에 대한 문제점들이 전부 해결된다고 하더라도 software implementation이 쉽지 않다.

Solution

- Practical Bayesian Optimization of Machine Learning Algorithms (NIPS 2012)
  - 여러 실험을 통해 Matern 5/2 kernel이 가장 좋은 결과
  - acquisition function도 EI로 고정
  - GP의 hyperparameter들을 Bayesian approach를 통해 acquisition function을 hyperparameter에 대해 marginalize한 후, MCMC 사용

Solution Matern 5/2 kernel



Squared-exponential function는 복잡한 모델을 표현하기에 너무 'smooth'한 function만 estimate할 수 있다는 단점

해결하기 위해 Matern사용 hyper parameter로 5, 2 사용

$$K_{M52}(x,x') = \theta_0 \left(1 + \sqrt{5r^2(x,x')} + \frac{5}{3}r^2(x,x')\right) \exp\left\{-\sqrt{5r^2(x,x')}\right\}.$$

$$r^2(x,x') = \sum_{d=1}^{D}(x_d - x'_d)^2/\theta_d^2.$$

Solution hyper parameter 튜닝

- GP의 hyper parameter는 θ_0, θ_d d+1개, constant mean function의 값 m, noise σ 총 d + 3 개

  ex) random forest의 경우, hyperparameter는 n_estimators, criterion, max_depth, min_samples_leaf로 d=4, 총 7개의 hyper parameter

- hyperparameter를 완전하게 Bayesian으로 처리하기 위하여 모든 hyperparameter θ (d+3 dimensional vector)에 대해 acquisition function을 marginalize

- integrated acquisition function

$$\hat{a}(x;\{x_n,y_n\}) = \int a(x;\{x_n,y_n\},\theta)p(\theta\|\{x_n,y_n\})_{n=1}^{N})d\theta.$$

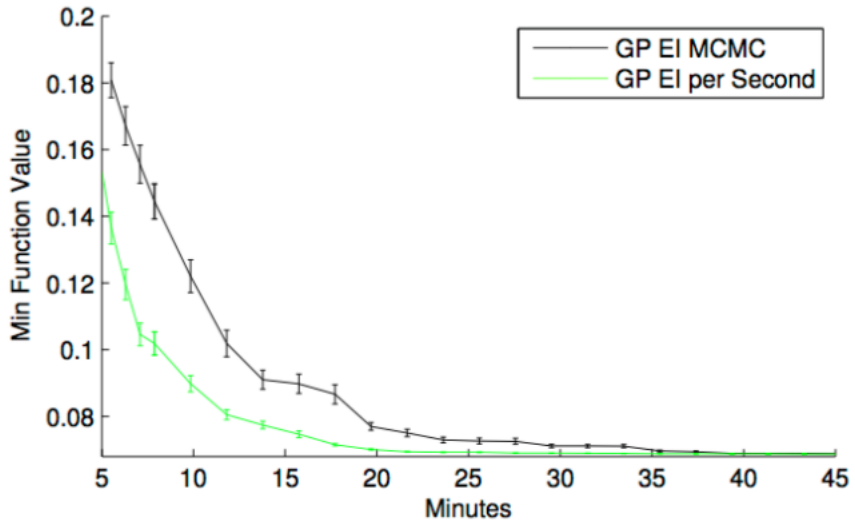
- integrated acquisition function의 Monte Carlo estimatation을 구하는 것이 가능
  (sampling을 통해 얻은 여러 hyperparameter들에 대해 EI를 전부 구한 다음, 그것들을 사용해 expectation 계산 => integrated EI

Solution hyper parameter 튜닝

Expected Improvement per second

위에서 구한 integrated EI를 사용한다고 하더라도, 아직 몇 가지 문제점들이 남아있다. 그 중 하나는, 모든 hyperparameter에 대해 실험 시간이 똑같지 않다는 점이다. 예를 들어 deep learning layer가 2인 것과 500인 것은 실험 시간의 차이가 어마어마하다. 따라서 실제로는 가장 최소한의 시행을 통해 optimization을 진행한다고 하더라도, 실제 소요 시간은 엄청 클 수도 있는 것이다. 이 논문은 그런 문제를 해결하기 위해, 필요한 경우 EI per second 라는 새로운 acquisition function을 제안한다.

아마도 NIPS 논문이 page limitation이 빡빡해서 그런지 정확한 formulation은 나와있지 않지만, 요점은 objective function f(x) 말고도, duration function c(x) 라는 것을 따로 정의한 다음, 이 함수를 사용해 'cost'를 모델링하는 것이다. c(x)도 GP라고 assume하는 것 같은데, c(x)와 f(x)가 independent하다고 가정하면 쉽게 acquisition function을 구할 수 있는 모양이다. 아래 실험결과에서도 볼 수 있듯, 오히려 실제 실행 시간의 관점에서는 EI per second가 더 빠른 것을 알 수 있다.



Result

4.4 Convolutional Networks on CIFAR-10

Neural networks and deep learning methods notoriously require careful tuning of numerous hyper-parameters. Multi-layer convolutional neural networks are an example of such a model for which a thorough exploration of architechtures and hyperparameters is beneficial, as demonstrated in Saxe et al. [21], but often computationally prohibitive. While Saxe et al. [21] demonstrate a methodology for efficiently exploring model architechtures, numerous hyperparameters, such as regularisation parameters, remain. In this empirical analysis, we tune nine hyperparameters of a three-layer convolutional network [22] on the CIFAR-10 benchmark dataset using the code provided [5]. This model has been carefully tuned by a human expert [22] to achieve a highly competitive result of 18% test error on the unaugmented data, which matches the published state of the art result [23] on CIFAR-10. The parameters we explore include the number of epochs to run the model, the learning rate, four weight costs (one for each layer and the softmax output weights), and the width, scale and power of the response normalization on the pooling layers of the network.

We optimize over the nine parameters for each strategy on a withheld validation set and report the mean validation error and standard error over five separate randomly initialized runs. Results are presented in Figure 6 and contrasted with the average results achieved using the best parameters found by the expert. The best hyperparameters found by the GP EI MCMC approach achieve an error on the *test set* of 14.98%, which is over 3% better than the expert and the state of the art on CIFAR-10. The same procedure was repeated on the CIFAR-10 data augmented with horizontal reflections and translations, similarly improving on the expert from 11% to 9.5% test error. To our knowledge this is the lowest error reported, compared to the 11% state of the art and a recently published 11.21% [24] using similar methods, on the competitive CIFAR-10 benchmark.

Reference

1. Practical Bayesian Optimization of Machine Learning Algorithms (NIPS2012) (https://proceedings.neurips.cc/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html)

2. http://sanghyukchun.github.io/99/

3. https://greeksharifa.github.io/bayesian_statistics/2020/07/12/Gaussian-Process/

4. http://dmqm.korea.ac.kr/activity/seminar/285

5. https://github.com/fmfn/BayesianOptimization