사용한 폰트: Noto Sans KR
(Light, Regular, Bold)

교재 05-1, 05-2장

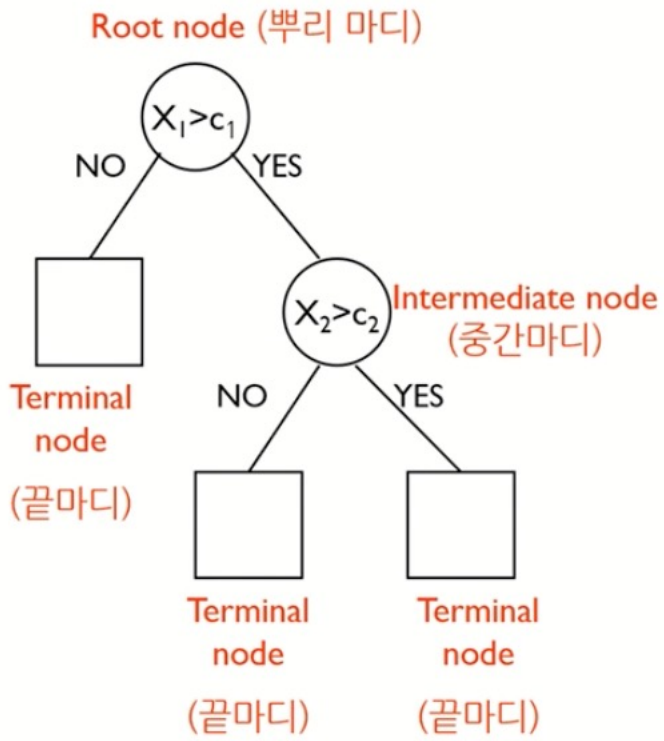
트리 알고리즘

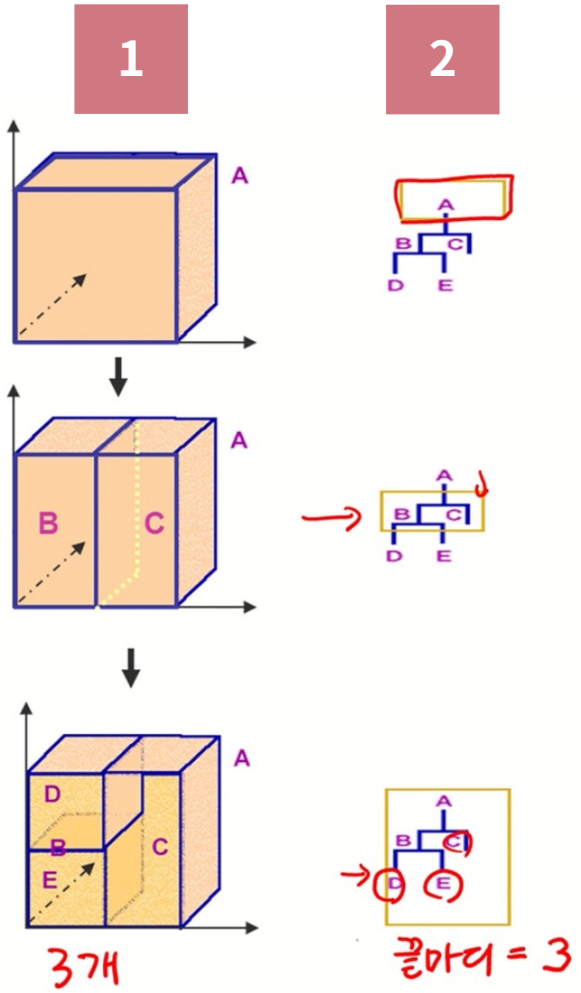
결정 트리, 교차 검증과 그리드 서치

2022.11.16 (수)

2조 강유정 이두진 이주석 임홍주

01. 결정 트리





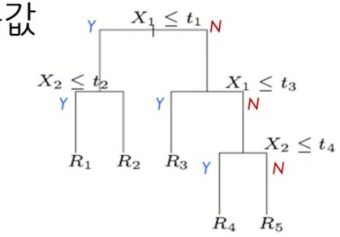
결정 트리의 특징:

<u>이진 분할을 쉽게 표현할 수 있음</u>

- 1번은 변수가 3개 있어야 표현 가능 (3차원)
- 반면 2번은 변수 개수가 늘어나도 차원에 관계없이 항상 표현 가능하며 해석력이 좋다는 장점이 있음

01. 결정 트리 _ 회귀 (예측 나무 모델)

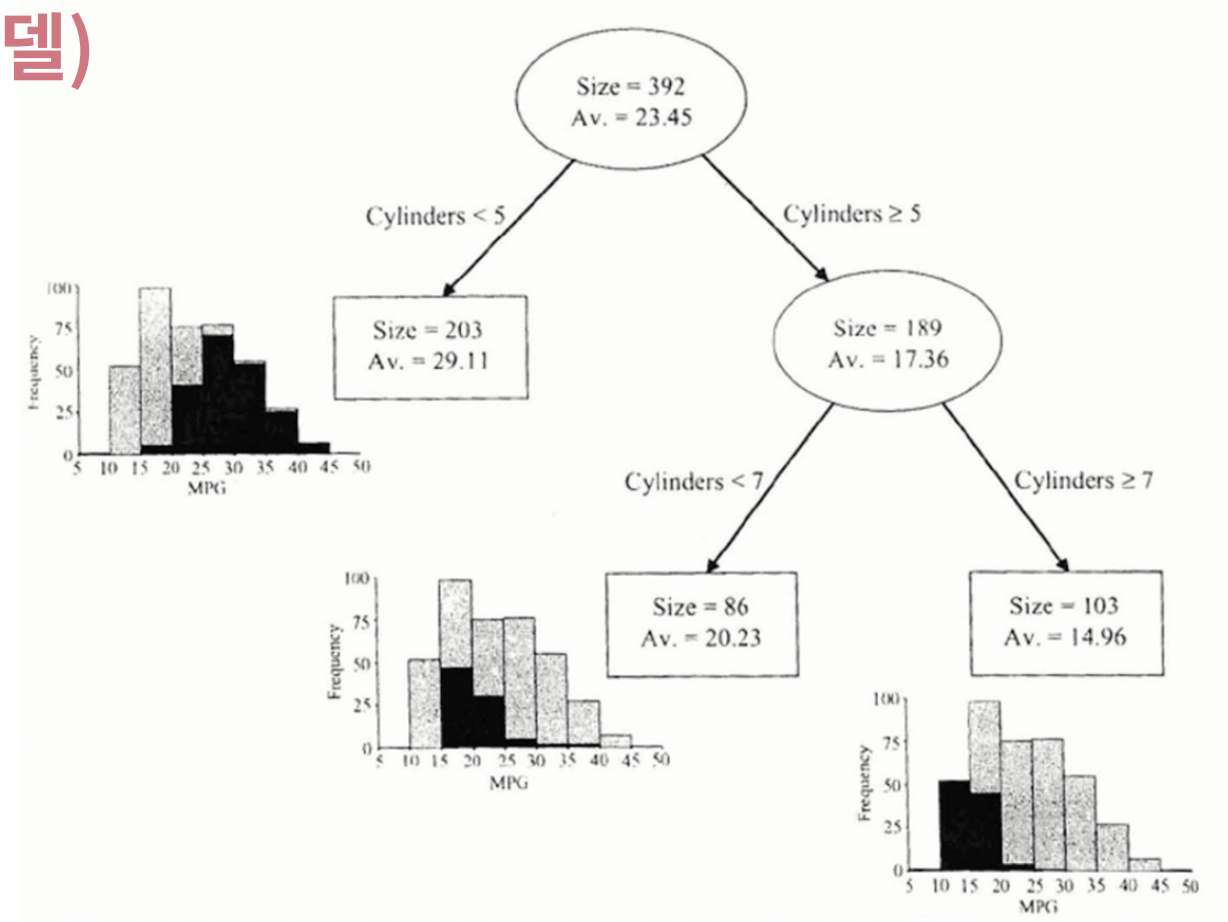
- C_m: 회귀나무모델로 부터 예측한 R_m 부분의 예측값



$$\hat{f}(x) = \sum_{m=1}^{5} c_m I\{(x_1, x_2) \in R_m\}$$
$$= c_1 I\{(x_1, x_2) \in R_1\} + c_2 I\{(x_1, x_2) \in R_2\} + c_3 I\{(x_1, x_2) \in R_3\} + c_4 I\{(x_1, x_2) \in R_4\} + c_5 I\{(x_1, x_2) \in R_5\}$$

$= c_1 \cdot 0 + c_2 \cdot 0 + c_3 \cdot 1 + c_4 \cdot 0 + c_5 \cdot 0$

$= c_3$



- I : Indicator Function (조건이 거짓일 때 0, 조건이 참일 때 1을 출력하는 이진 분류 함수)

- 처음에 Y값 평균은 23 → 끝마디의 평균은 각각 29, 20, 14
- 히스토그램으로 확인해보면 실제로 동질적인, 균일한 집단끼리 모였음

01. 결정 트리 _ 회귀 (예측 나무 모델)

1

- 최상의 분할은 다음 비용함수(cost function)를 최소로 할 때 얻어짐

$$\min_{C_m} \sum_{i=1}^{N} (y_i - f(x_i))^2$$

$$= \min_{C_m} \sum_{i=1}^{N} (y_i - \sum_{m=1}^{M} c_m I(x \in R_m))^2$$

$$\hat{c}_m = ave(y_i | x_i \in R_m)$$

- 각 분할에 속해 있는 y값들의 평균으로 예측했을 때 오류가 최소

2

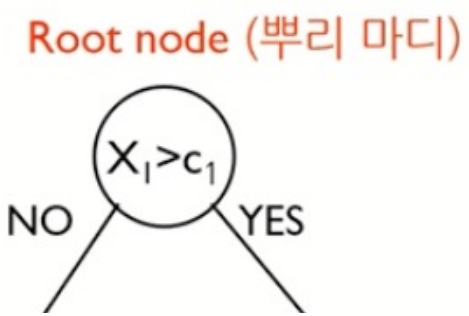
- 분할변수(j)와 분할점(s)은 어떻게 결정할까?

$$R_1(j,s) = \{x | x_j \leq s\}$$
$$R_2(j,s) = \{x | x_j > s\}$$

$$\underset{j,s}{\text{argmin}} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right]$$

$$= \underset{j,s}{\text{argmin}} \left[\sum_{x_i \in R_1(j,s)} (y_i - \hat{c}_1)^2 + \sum_{x_i \in R_2(j,s)} (y_i - \hat{c}_2)^2 \right]$$

$$\hat{c}_1 = ave(y_i | x_i \in R_1(j,s) \text{ and } \hat{c}_2 = ave(y_i | x_i \in R_2(j,s)$$



- 예를 들어 첫 번째 질문이 X1 > 2 라면,
  - 분할변수(j)는 X1 이므로 j=1, 분할점(s)=2
- 분할변수와 분할점 찾는 것은 grid search 원리와 같다
  - 가능한 경우를 모두 계산해보고 위의 식을 최소로 만드는 j와 s를 찾는다

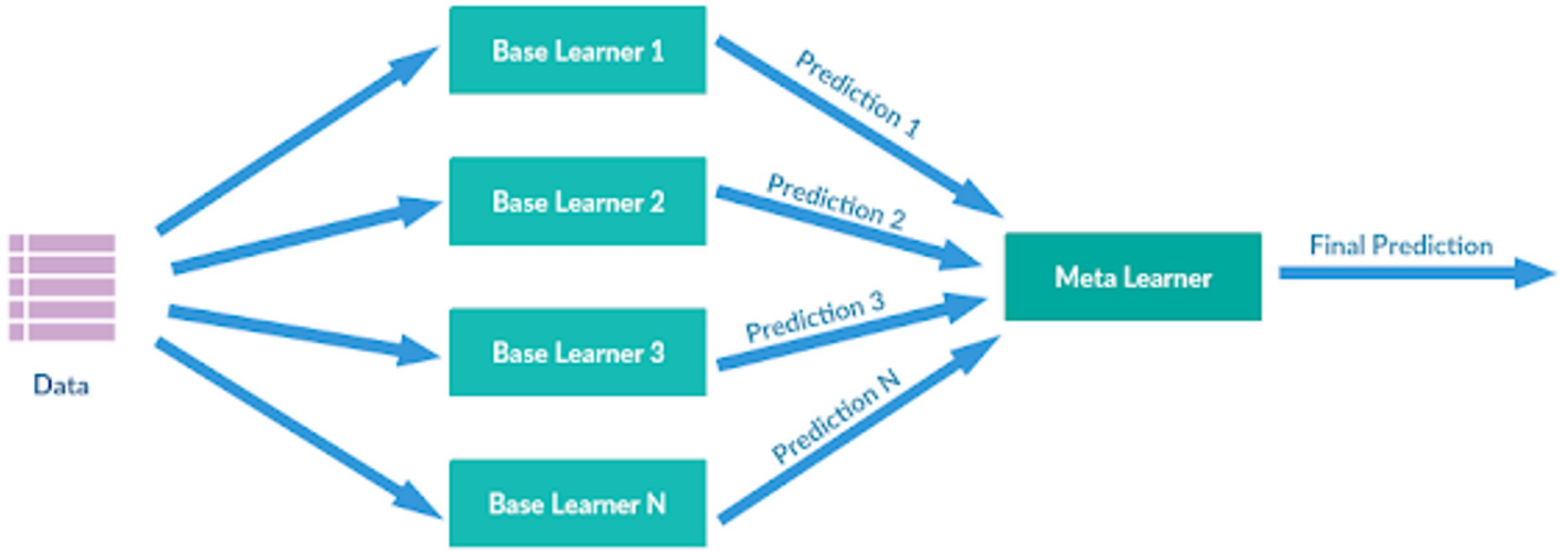
02. 교차 검증

앙상블 학습 기법

01 Bagging

02 Boosting

03 **Stacking**



- 개별 알고리즘의 예측한 데이터를 기반으로 다시 예측을 수행하는 방법
- 개별 모델(Base Learner)이 예측한 데이터를 다시 meta data set으로 사용해서 학습
- Base Learner들이 동일한 데이터 원본 데이터를 가지고 그대로 학습

➔ Overfitting 문제 발생 ➔ Cross Validation 방법 도입

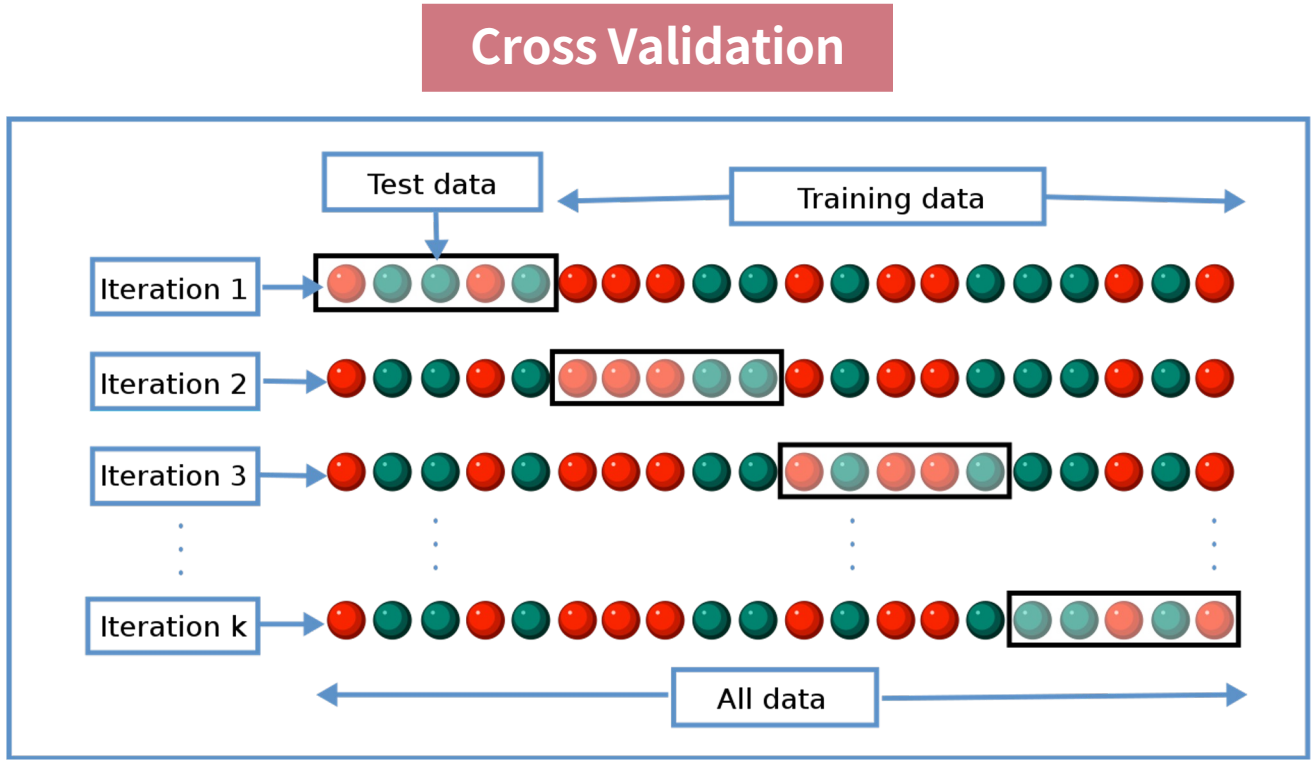
02. 교차 검증

앙상블 학습 기법

01 **Bagging**

02 **Boosting**

03 **Stacking**



- 전체 데이터 셋을 k개로 나누고 k 번의 평가를 실행하는데, 이때 test set을 중복 없이 바꾸어 가면서 평가
- 모든 데이터 셋을 평가에 활용할 수 있어, 데이터 편중을 막아 overfitting 방지
- Iteration 횟수가 많기 때문에 모델 훈련/평가 시간이 오래 걸림

02. 교차 검증

앙상블 학습 기법

01 Bagging

02 Boosting

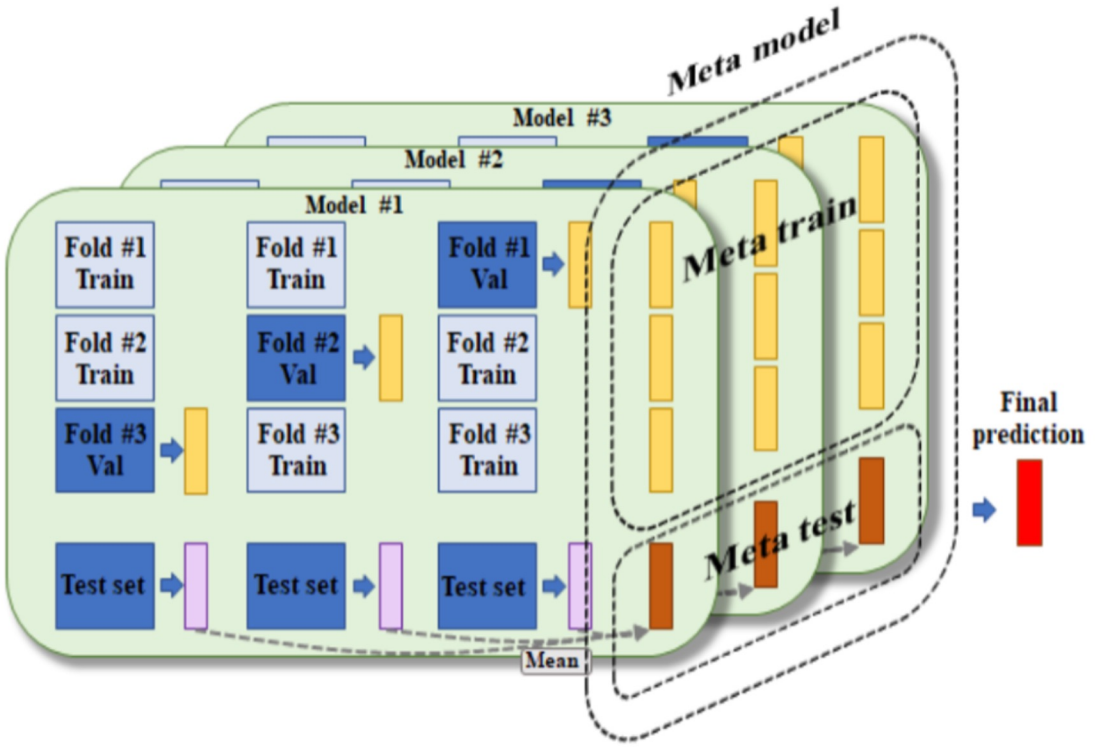
03 Stacking

Cross Validation CV 세트 기반의 Stacking

① Train set을 N 개의 fold로 나눈다. (3개의 fold로 나누었다 가정)

② 2개의 fold를 학습을 위한 데이터 fold로, 1개의 fold를 검증을 위한 데이터 fold로 사용

③ 위의 2개의 fold를 이용해 개별 모델을 학습, 1개의 검증용 fold로 데이터를 예측 후 결과를 저장



④ 위 로직을 3번 반복(학습, 검증용 fold를 변경해가면서) 후

Test set에 대한 예측 값의 평균으로 최종 결괏값 생성

⑤ 위에서 생성된 최종 예측 결과를 메타 모델에 학습 및 예측 수행

03. 그리드 서치

- Grid search란? 모델에게 가장 적합한 (우수한 성능을 보이는) 하이퍼 파라미터 찾기

```
from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(model, n_jobs=-1)
```

- 사용할 코어 수를 지정
- 사용하는 CPU 코어 개수에 비례해서 속도도 빨라짐
- 코랩 CPU core개수는 2개이므로 n_jobs=-1 과 n_jobs=2 는 같다

CPU-only VMs

Parameter	Google Colab	Kaggle Kernel
No. **CPU Cores**	2	4
CPU Family	Haswell	Haswell
Available RAM	12GB (upgradable to 26.75GB)	16GB
Disk Space	25GB	5GB

2행 더 보기

kazemnejad.com › blog › how_to_do_deep_learning_res...

How to do Deep Learning research with absolutely no GPUs ...

장점

- 주어진 공간 내에서 가장 좋은 결과를 얻을 수 있음

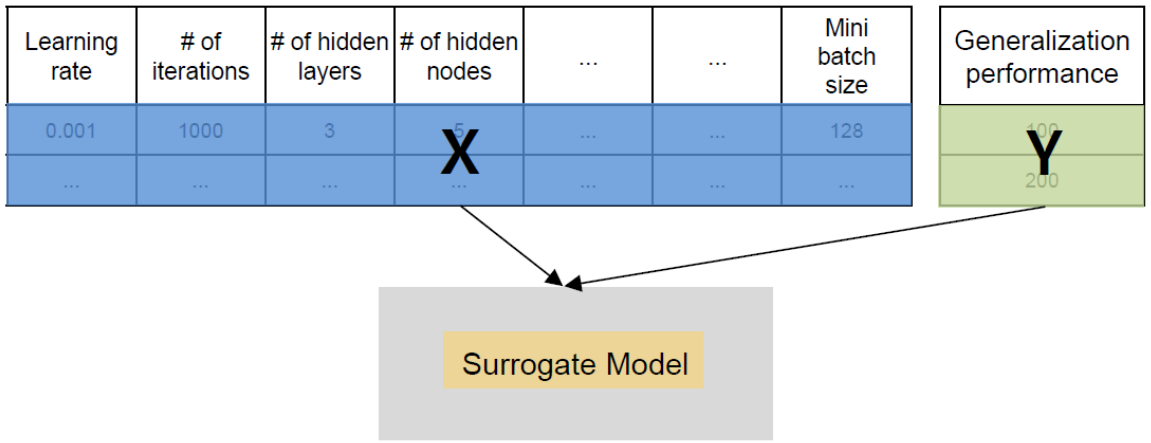
단점

- 시간이 오래 걸림
- 입력하지 않은 값이 최적 파라미터일 수도 있음

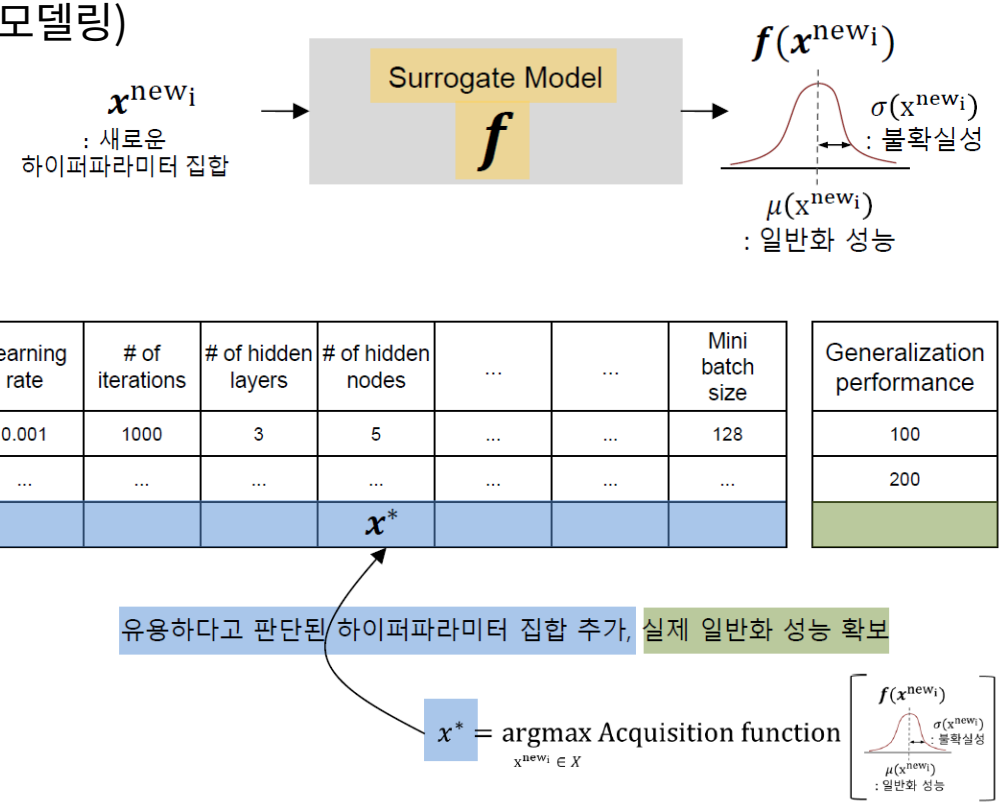
03. 그리드 서치 _ 베이즈 최적화

- GridSearch의 단점: 모든 하이퍼파라미터 후보들에 대한 일반화 성능을 확인하기 때문에 시간이 너무 오래 걸림
- RandomSearch의 단점: GridSearch에 비해 시간은 적게 걸리지만, 말 그대로 "랜덤"하게 몇 개만 뽑아서 확인해보기에 정확도가 다소 떨어질 수 있음

➔ 베이즈 최적화: 효율적으로 최적값을 찾아냄 (Surrogate Model과 Acquisition Function 이용해 “새로운 하이퍼파라미터가 계속 추가되는 하이퍼파라미터 집합”과 “일반화 성능”의 관계를 모델링)



하이퍼파라미터 집합과 일반화 성능의 관계를 모델링



감사합니다