

최적화란? 여러 가능한 의사결정 중, 주어진 조건을 만족하며 가장 좋은 결과를 만드는 선택을 찾는 과정  
(Optimization)

## 개념정리

- ① **최적화문제**: 내가 바꿀 수 있는 값을 어떻게 정해야 결과가 가장 좋아지는가?
- ② **Decision Variable (의사결정 변수)**: 직접 바꿀 수 있는 값
- ③ **Objective function (목적함수)**: 특정 Decision Variable을 선택했을 때 그 결과를 수치로 평가하는 함수
- ④ **Cost function (비용함수)**: Objective function 중 작을수록 좋은 값으로 정의한 경우 (예: 착지 오차, 에너지 비용)
- ⑤ **Loss function (손실함수)**: 머신러닝에서 주로 사용하는 Cost/Objective 함수의 형태이며, 예측이 틀린 정도 (일반적으로 최소화함)
- ⑥ **Decision Space (의사결정 공간)**: Decision Variable 이 가질 수 있는 모든 가능값의 집합 (변수 1개 - 선, 2개 - 평면)
- ⑦ **Optimization Landscape (최적화지형)**: Decision Variable 에 따라 Objective function 값이 어떻게 변하는지를 나타낸 형태
- ⑧ **Gradient (기울기)**: 현재 위치에서 목적함수 값이 가장 빠르게 증가하는 방향과 변화 정도 (-Gradient: Gradient의 반대 방향)  
목적함수를 최소화할 때 이동해야 하는 방향
- ⑨ **Step size**: 한 번 업데이트 할 때 얼마나 이동할지 정하는 값
- ⑩ **Learning Rate (학습률)**: 업데이트에서 Step size를 부르는 용어. 너무 작으면 느리고, 너무 크면 최적점을 지나칠 수 있다.
- ⑪ **Update**: 현재 Decision Variable을 Gradient를 이용해 새로운 값으로 변경.
- ⑫ **Minimum (최솟값)**: Objective/Cost가 가장 작은 지점
- ⑬ **Convergence (수렴)**: 반복적인 Update 후 더 이상 값이 변하지 않는 상태 (Gradient가 0에 가까워지면 이동이 멈춤)
- ⑭ **Gene (유전자)**: 하나의 의사결정 단위
- ⑮ **Chromosome (염색체)**: 여러 Gene이 모인 하나의 완성된 후보 해
- ⑯ **population (개체군)**: 한 세대에 존재하는 여러 후보 해들의 집합
- ⑰ **Particle (입자)**: 하나의 후보 해 (ex. 드론 - kd, kp)
- ⑱ **Position (위치)**: 현재 입자가 가진 후보 해의 값 (ex. 현재의 kd, kp)

① Velocity (속도) : 입자가 다음 위치로 얼마나, 어느 방향으로 이동할지 나타내는 값

② pbest (personal best) : 각 입자가 지금까지 방문한 위치 중 가장 좋았던 위치

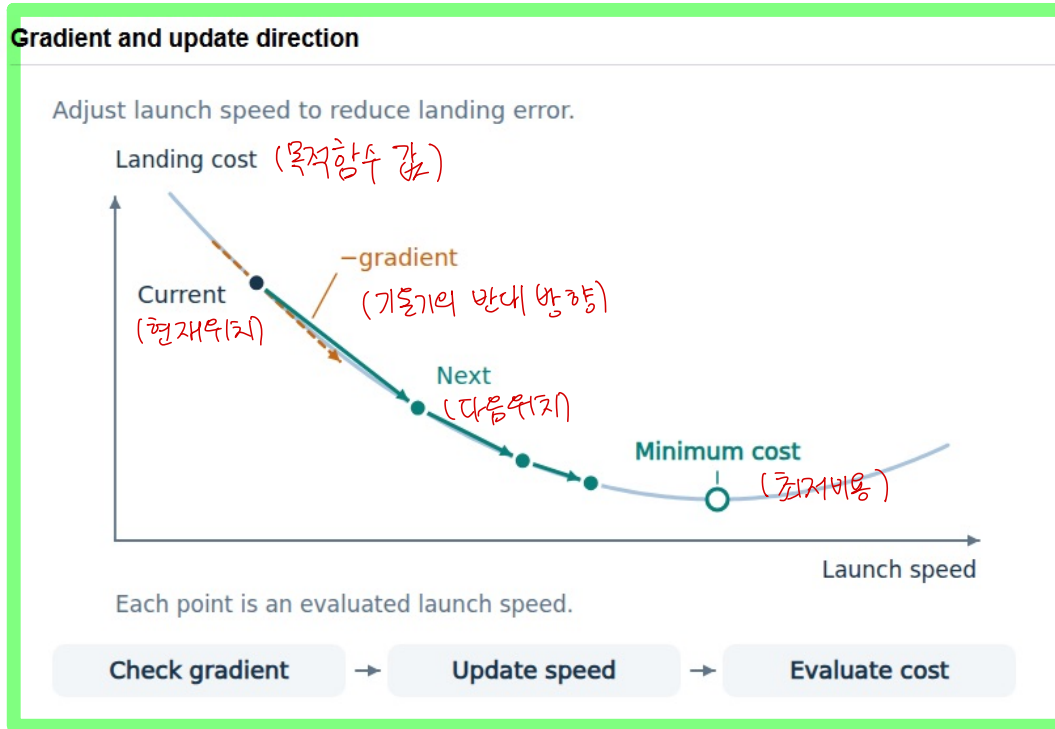
③ gbest (global best) : 전체 입자 중 현재까지 가장 좋았던 위치

④ Inertia (관성) : 입자가 원래 가던 방향을 어느정도 유지하려는 성질

⑤ Swarm (군집) : 함께 탐색하는 여러 입자 전체

# 1. Gradient Descent (경사하강법)

: 현재 위치의 Gradient를 계산하고,  $-Gradient$  방향으로 조금씩 이동하여 목적함수를 최소화하는 방법.



## 핵심 아이디어

- 현재 위치의 기울기 (Gradient)를 계산
- $-gradient$  방향으로 조금 이동 (Step size 만큼)
- 새로운 위치에서 목적함수 값을 다시 평가
- 이 과정을 반복하여 최저점에 도달

decision Variables 를 업데이트 할 때  
기존 값에  $-gradient * stepsize$  를  
더하는 방식으로 업데이트 함.

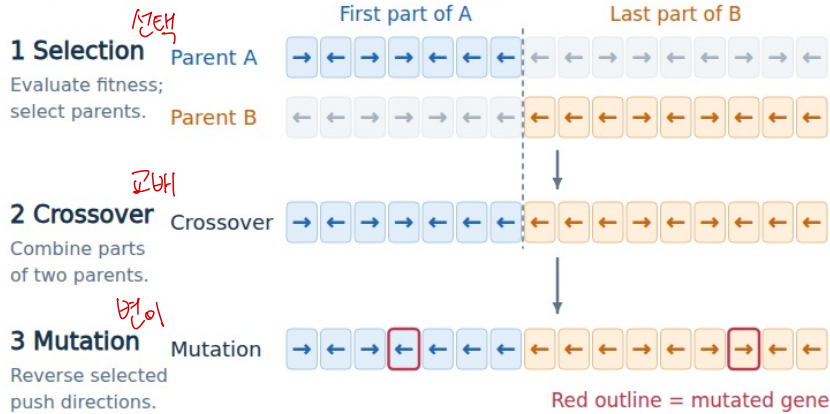
## 2. Genetic Algorithm (유전자 알고리즘)

: 여러 후보 해 중 성능이 좋은 해를 선택하고, 교배와 변이를 반복하여  
세대가 지날수록 더 좋은 해를 찾는 방법

### Selection, crossover, and mutation

Evolve push sequences to increase swing height.

One gene = push direction for 0.5 s (← left / → right)



### Genetic Algorithm (유전자 알고리즘)

: 여러 후보 해(밀기 동작)를 평가하고, 좋은 해를 선택·교배·변이하여  
세대가 지날수록 그네를 더 높게 올리는 방법 (진화 기반 탐색)

#### 문제 설정



- 하나의 유전자(gene)  
: 0.5초 동안의 밀기 방향 (← : 왼쪽, → : 오른쪽)
- 하나의 염색체(chromosome)  
: 여러 개의 유전자로 이루어진 전체 밀기 순서

#### 1 초기 후보 생성 (Initialization)

여러 개의 랜덤한 밀기 순서(후보 해)를 생성

후보 1: → → ← → ← ...  
후보 2: → ← → ← → ...  
후보 3: → ← → ← → ...  
후보 4: → ← → ← → ...  
...

#### 2 적합도 평가 (Evaluation)

각 후보로 그네를 타고 최고로 올라간 높이(적합도)를 측정

후보 1: 높이: 1.2 m  
후보 2: 높이: 2.8 m  
후보 3: 높이: 0.9 m  
후보 4: 높이: 1.5 m  
...

#### 3 선택 (Selection)

적합도가 높은(더 높이 올라간) 후보를 부모로 선택

더 높이 올라간 좋은 후보를 선택!  
Parent A (높이 2.8 m)  
Parent B (높이 2.3 m)  
(제거) (제거)

#### 4 교배 (Crossover)

두 부모의 밀기 순서를 섞어 새로운 자식 생성

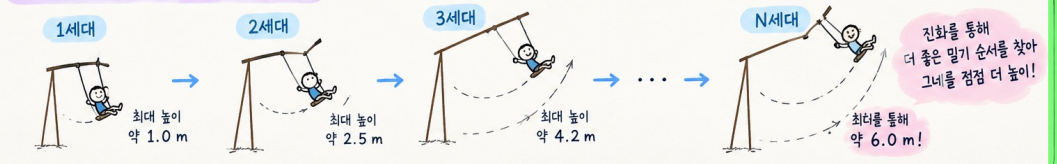
Parent A: → → ← → ← ...  
Parent B: → ← → ← → ...  
자식 (Child): → → ← → ← ...  
자로는 위치

#### 5 변이 (Mutation)

일부 유전자를 랜덤하게 변경하여 새로운 탐색이 가능하도록 함

자식: → → ← → ← ...  
랜덤하게 방향을 변경!  
6 다음 세대 생성  
새로 만든 자식들을 포함하여 다음 세대를 구성하고, 1~5 과정을 반복!  
세대가 지날수록 더 높은 그네를 타는 해를 발견!

#### 세대가 지날수록 점점 더 높이!



## 핵심 아이디어

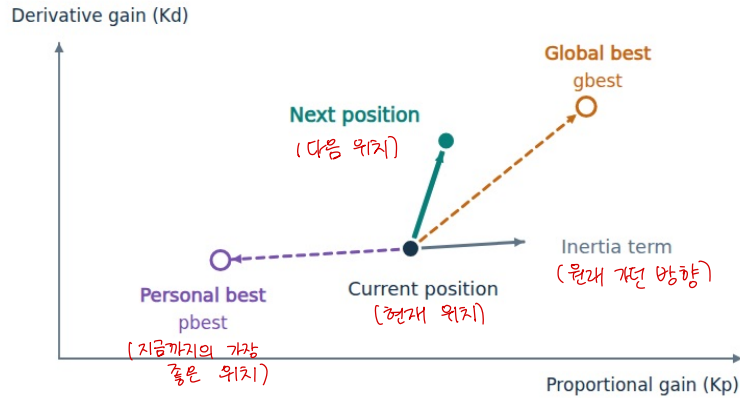
- 그네를 밀 수 있는 초기 후보 생성 → 그네 높이 평가 → 좋은 후보 선택 → 교배 → 변이 → 다음 세대 → 반복 → 최대 높이 지점 도출

### 3. Particle Swarm Optimization, PSO (입자군집 최적화)

: 여러 입자가 자신의 최고 위치와 전체 집단의 최고 위치를 참고하여  
이동을 반복하면서 최적해를 찾아가는 방법

#### Personal best and global best

Each point is a pair of drone controller gains.



Update position using inertia, personal best, and global best.

Repeat position updates and objective evaluations across the swarm.

#### Particle Swarm Optimization (PSO, 입자군집최적화)

: 여러 입자(후보 해)가 자신의 경험과 전체 무리의 경험을 참고하며 더 좋은 해를 함께 찾아가는 방법.

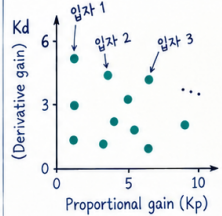
##### 문제 설정



- 입자(Particle) = 하나의 후보 해
- 한 입자의 값 =  $(K_p, K_d)$  한 쌍 (드론의 제어 계인)

##### 1 초기 입자 생성 (Initialization)

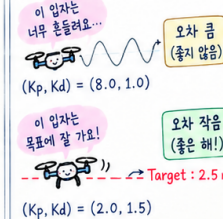
여러 후보 해를 무작위로 배치!



$(K_p, K_d)$  공간에 여러 입자를 무작위로 배치하여 탐색을 시작!

##### 2 적합도 평가 (Evaluation)

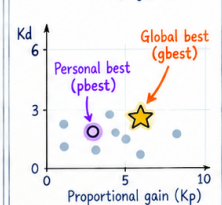
각 입자로 드론을 제어해 보고 성능(비용/오차)을 계산!



목표에 더 잘 맞을수록 더 좋은 해 (높은 적합도)!

##### 3 pbest와 gbest 찾기

- pbest = 각 입자가 지금까지 찾은 가장 좋은 위치
- gbest = 모든 입자 중 가장 좋은 위치



각 입자는 자신의 경험을 기억하고, 무리의 최고 경험을 공유해요!

##### 4 위치 업데이트

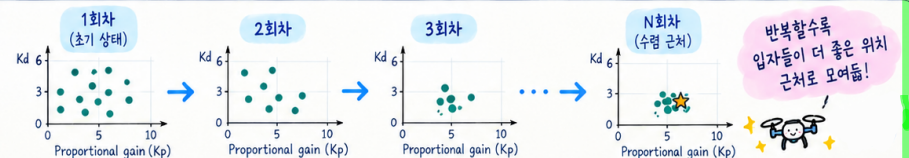
입자는 관성 + pbest 방향 + gbest 방향을 함께 고려해 다음 위치로 이동!



이 세 가지 방향을 합쳐서 다음 위치와 속도를 계산해요!

##### 5 반복 (Iteration)

평가와 업데이트를 반복하면, 입자들이 점점 더 좋은 위치 근처로 모여듭니다!



반복할수록 입자들이 더 좋은 위치 근처로 모여둡니다!

### 핵심 아이디어

- 각 입자는 자신의 경험 (pbest)과 무리의 경험 (gbest)을 참고
- 원래 가던 방향 (inertia)도 함께 고려하여 새로운 위치로 이동
- 각 위치의 성능을 평가하고, pbest와 gbest를 업데이트
- 이 과정을 반복하면 입자들이 좋은 위치로 모여감.