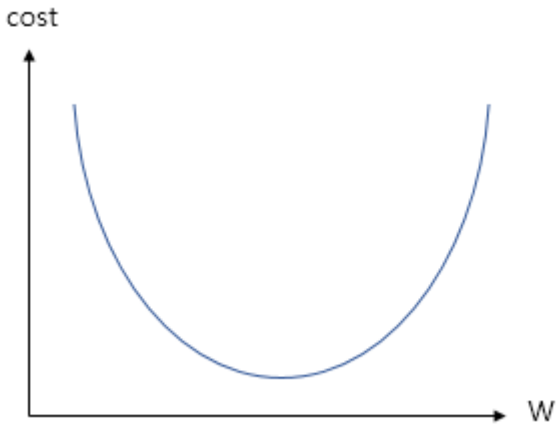
3_What is gradient descent

- 가설 : $H(x) = Wx + b$
- 그러나 기우
- cost = torch.mean((hypothesis - y_train) ** 2)
- W가 커지면 cost의 값도 커지고, 작아져도 커짐. (˙꒳˙ 제곱)



W와 cost의 관계를 그래프로 표현

- cost function을 최소화하려면 접선의 기울기가 0인 W를 찾아야함.

$$cost(W, b) = \frac{1}{m}\sum_{i=1}^{m}\left(H(x^{(i)}) - y^{(i)}\right)^2$$

- 기울기 = $\frac{\partial cost(W)}{\partial W}$ = $cost(W) = \frac{2}{m}\sum_{i=1}^{m}\left(W(x^{(i)}) - y^{(i)}\right)x^{(i)}$
- 기울기가 음수일 때 : W의 값이 증가
- 기울기가 양수일 때 : W의 값이 감소

- $W := W - \alpha\frac{\partial}{\partial W}cost(W)$

  α = learning rate

  learning rate는 W의 값을 변경할 때 얼마나 크게 변경할지를 결정함.

```
gradient = 2 * torch.mean((W * x_train - y_train) * x_train)
lr = 0.1
w -= lt * gradient
```

- full code

```
# 데이터
x_train = torch.FloatTensor([[1], [2], [3]])
y_train = torch.FloatTensor([[1], [2], [3]])

# 모델 초기화
W = torch.zeros(1, requires_grad=True)

# lr 정의
lr=0.01

nb_epochs = 1000
for epoch in range(nb_epochs + 1):

    # H(x) 계산
    hypothesis = x_train * W

    # cost, gradient 계산
    cost = torch.mean((hypothesis - y_train) ** 2)
    gradient = torch.sum((W * x_train - y_train) * x_train) # (2/m) 은 어디로 사라짐 ????

    # gradient로 H(x) 개선
    w -= lt * gradient

print(W, cost)
```