# Momentum

Momentum-based gradient descent is an optimization algorithm that accelerates the convergence of gradient descent by incorporating the concept of momentum from physics. It's designed to overcome some of the limitations of the standard gradient descent algorithm, particularly its tendency to oscillate or get stuck in local minima.

## Steps :

1. **Initialization:**
    - Define the learning rate (lr) and the momentum coefficient (beta).
    - Initialize the weights (weights) for your model (typically with zeros).
    - Initialize the velocity (v), which is a vector with the same dimensions as weights. You can start it with zeros as well.

2. **Forward Pass:**
    - For each data point (or mini-batch):
        - Pass the data point(s) through your model to compute predictions.

3. **Error Calculation:**
    - Calculate the error between the model's predictions and the actual target values for each data point in the current batch.

4. **Gradient Calculation:**
    - Calculate the negative gradient of the loss function with respect to the weights using the errors from the current batch.

5. **Velocity Update:**
    - Update the velocity using the following formula:

$$v_t = \beta * v_(t-1) - lr * grad$$

6. **Weight Update:**
    - Update the weights using the following formula:

$$w_t = w_(t-1) - lr * v_t$$

    This update incorporates the accumulated momentum from the velocity term.

## Benefits of Momentum:

- Faster convergence compared to standard gradient descent, especially for noisy or complicated loss functions.
- Can help escape local minima by accumulating momentum and potentially overcoming shallow valleys.

## Drawbacks of Momentum:

- Requires tuning the hyperparameter beta.
- Might not guarantee convergence to the global minimum in all cases.
- **Overshooting and Oscillations:**
    - If the momentum coefficient is too high, the algorithm may overshoot the minimum and oscillate around it.
    - Balancing momentum to avoid overshooting while maintaining acceleration is challenging.