# Gradient Descent with Momentum

1. **Update the velocity**:

where:

* + ( v ) is the velocity (momentum term).
  + ( beta ) is the momentum hyperparameter (typically between 0 and 1).
  + ( dW ) is the gradient of the loss with respect to the weights.

1. **Update the weights**:

W = W - learning\_rate \* v

# RMSprop

1. **Update the exponentially weighted average of the squared gradients**:

where:

* + SdW is the exponentially weighted average of the squared gradients.
  + dW is the gradient of the loss with respect to the weights.

1. **Update the weights**:

where:

* + ε is a small constant added for numerical stability (e.g., )

# **Adam optimization algorithm**

The **Adam optimization algorithm** (short for **Adaptive Moment Estimation**) is a popular optimization technique used in training deep learning models. It combines the advantages of two other methods: **Momentum** and **RMSprop**.

[Adam Optimization Algorithm](https://www.coursera.org/learn/deep-neural-network/lecture/w9VCZ/adam-optimization-algorithm?trk_ref=coach_copy)

Here are the key formulas used in the **Adam optimization algorithm**:

1. **Initialization**:
2. **Gradient Calculation**:
   * Compute gradients ( dw ) and ( db ) using the current mini-batch.
3. **Update Rules**:
   * **Momentum Update**:
   * **RMSprop Update**:
4. **Bias Correction**:
5. **Weight Update**:

### Hyperparameters:

* α - learning rate
* - typically set to 0.9
* - typically set to 0.999
* ε - a small constant (often )

# **Learning rate decay**

Alpha = 1 / (1 + decay\_rate \* epoch\_num) \* Alpha\_0

In this formula:

* **Alpha** is the current learning rate.
* **decay\_rate** is a value you choose to control how quickly the learning rate decreases.
* **epoch\_num** is the number of times the model has gone through the training data.
* **Alpha\_0** is the initial learning rate.

There are several types of learning rate decay methods you can use. Here are a few common ones:

1. **Exponential Decay**:
   * The learning rate decreases exponentially over time.
   * Formula:

Alpha = Alpha\_0 \* decay\_rate^epoch\_num

* + Here, **decay\_rate** is a value less than 1 (e.g., 0.95).

1. **Step Decay**:
   * The learning rate decreases by a fixed amount after a certain number of epochs.
   * For example, you might halve the learning rate every 10 epochs.
2. **Polynomial Decay**:
   * The learning rate decreases following a polynomial function.
   * Formula:

Alpha = Alpha\_0 \* (1 - epoch\_num / total\_epochs)^power

* + Here, **power** controls the rate of decay.

1. **Manual Decay**:
   * The learning rate is adjusted manually based on the model's performance during training.
   * This requires monitoring the training process and making adjustments as needed.