Learning Rates and Weight Initialization

Foundations of Effective Neural Network Training

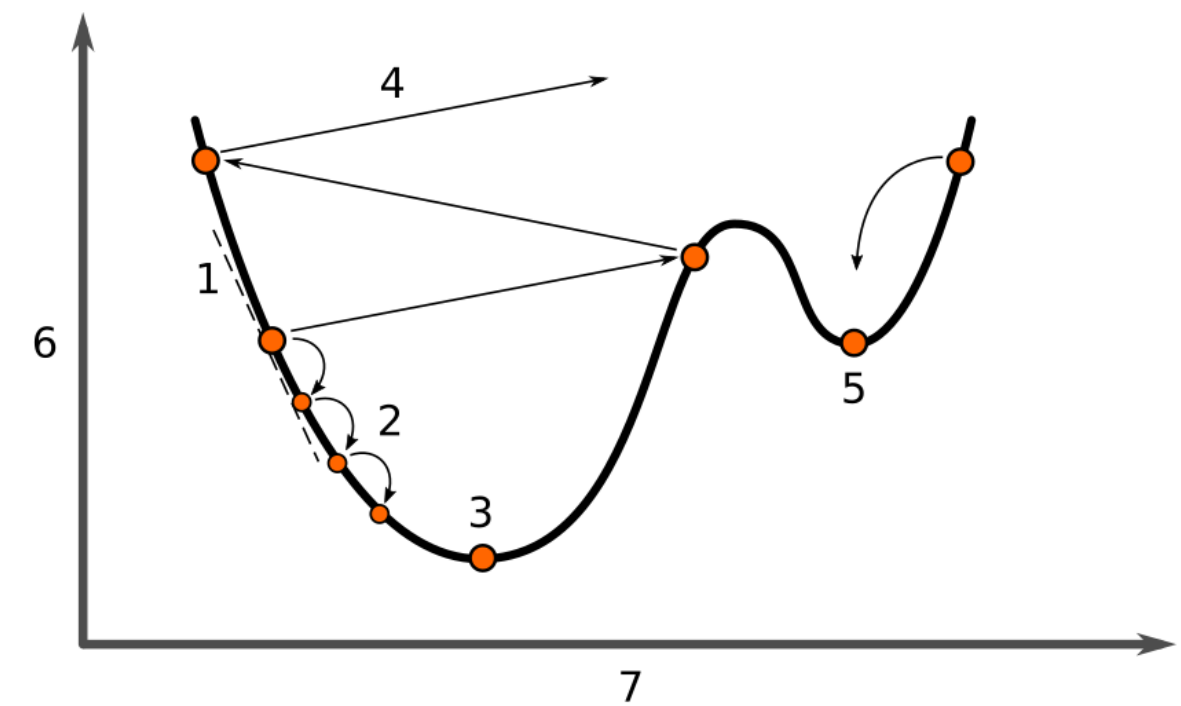
Sina Moghimi, Autumn 2024

Outline

- Learning Rate Fundamentals
- Learning Rate Impact
- Weight Initialization Basics
- Common Initialization Strategies
- Vanishing Gradient Problem
- Exploding Gradient Problem
- Initialization Techniques

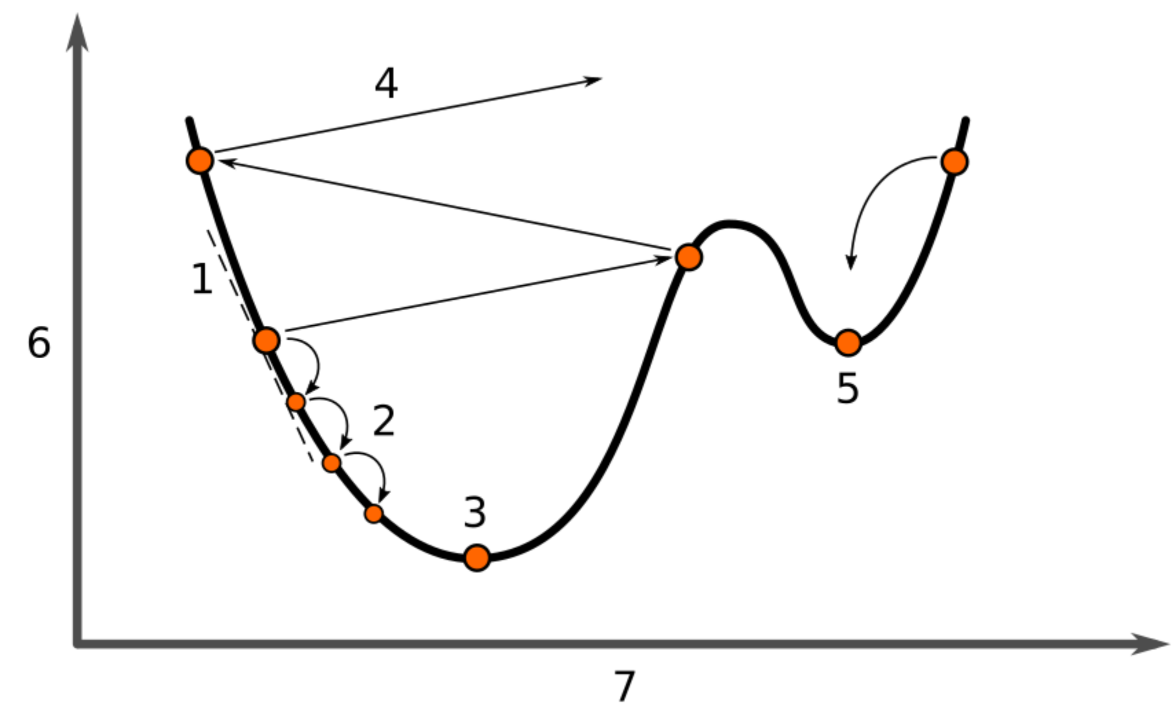
Learning Rate Fundamentals

- **Learning Rate (η)** is a critical hyperparameter in optimization
    - It matters how you approach the local minima
    - **Small Learning Rate**
        - Slow Convergence
        - Can get stuck in minimas like saddle points
    - **Large Learning Rate**
        - Fast Convergence, but might overshoot the desired minimum
        - Can cause Loss function to oscillate/diverge
    - **Optimal Learning Rate**
        - Balances the speed and stability



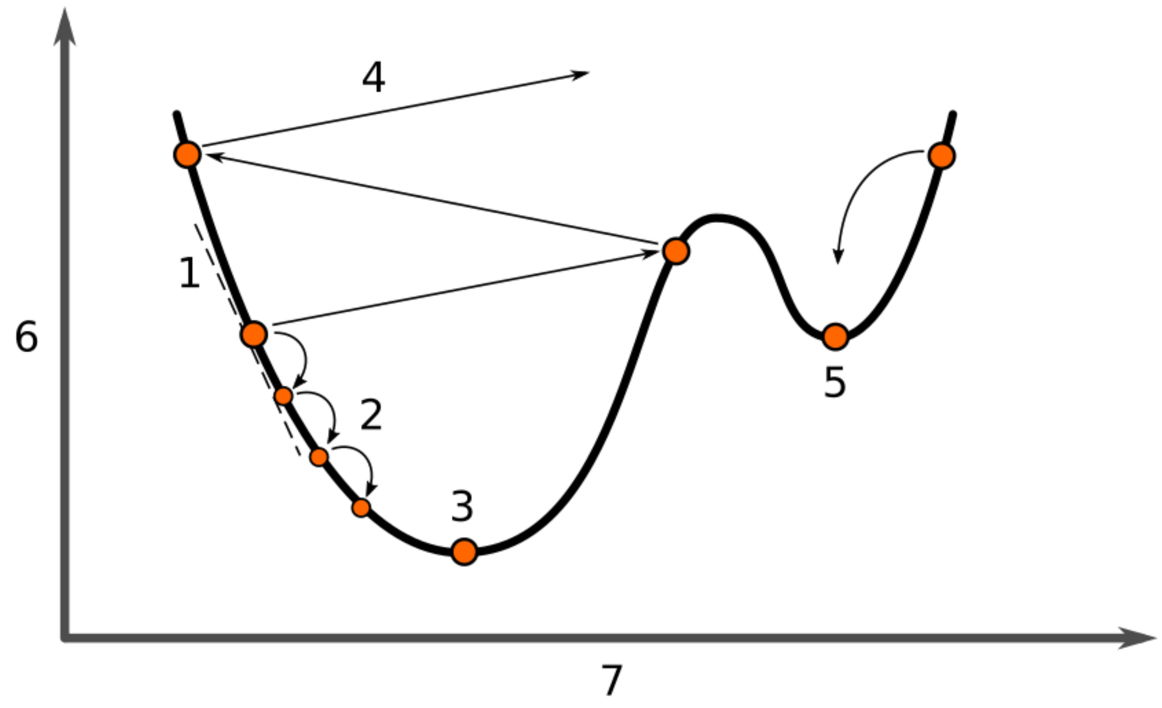
Learning Rate Fundamentals

- **Challenges**
  - Choosing the right value
  - Dynamic Adjustments
    - Learning Rate schedules(i.e, step decay, exponential decay)
  - Sensitivity to scale
    - Different features or layers may need different learning rates
    - Feature Scaling
  - Impact of loss surface
    - A highly non-convex loss surface with multiple local minima and saddle points complicates the choice of learning rate.



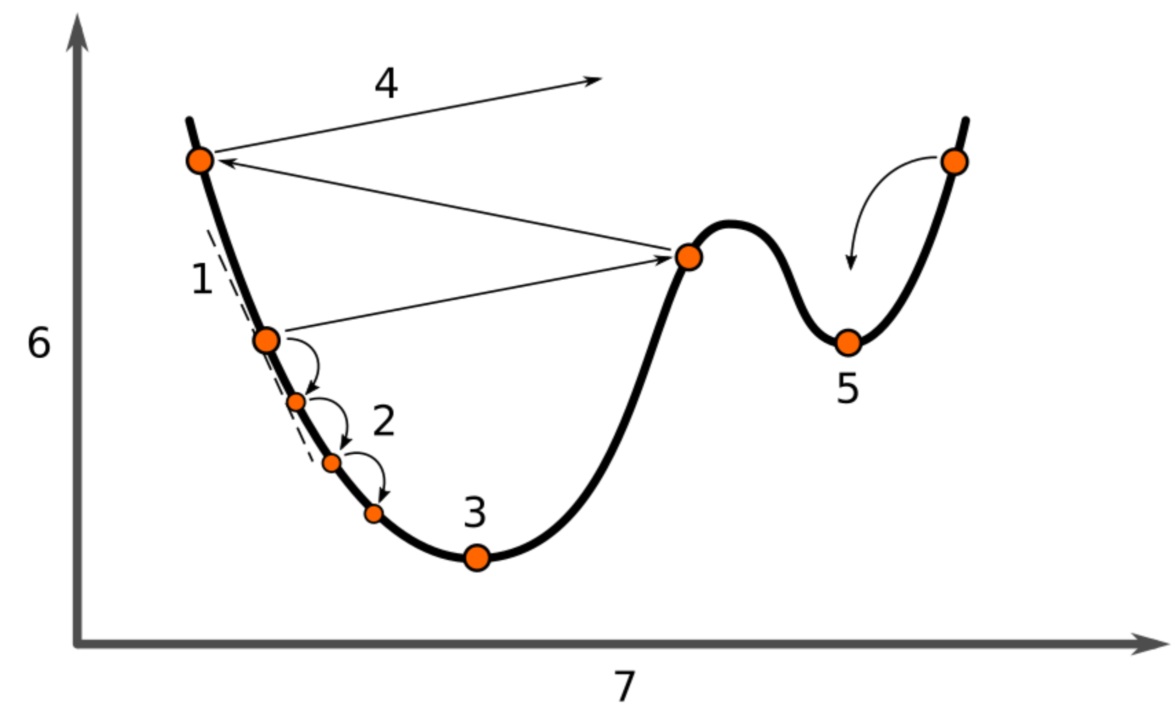
Weight Initialization Basics

- **Should we initialize the weights randomly?!**



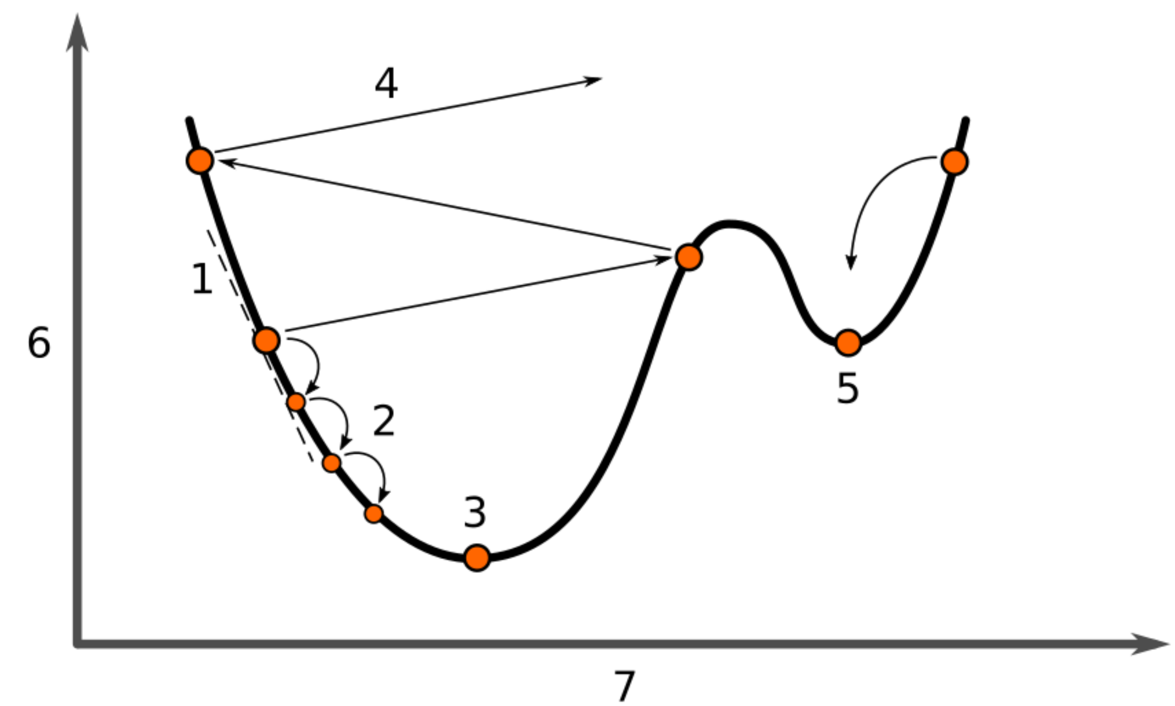
Weight Initialization Basics

- Process of assigning initial values to weights
  - It matters where we start from
- Proper Initialization is crucial
  - Ensure Effective Signal Propagation
  - Improve convergence speed
  - Prevent symmetry



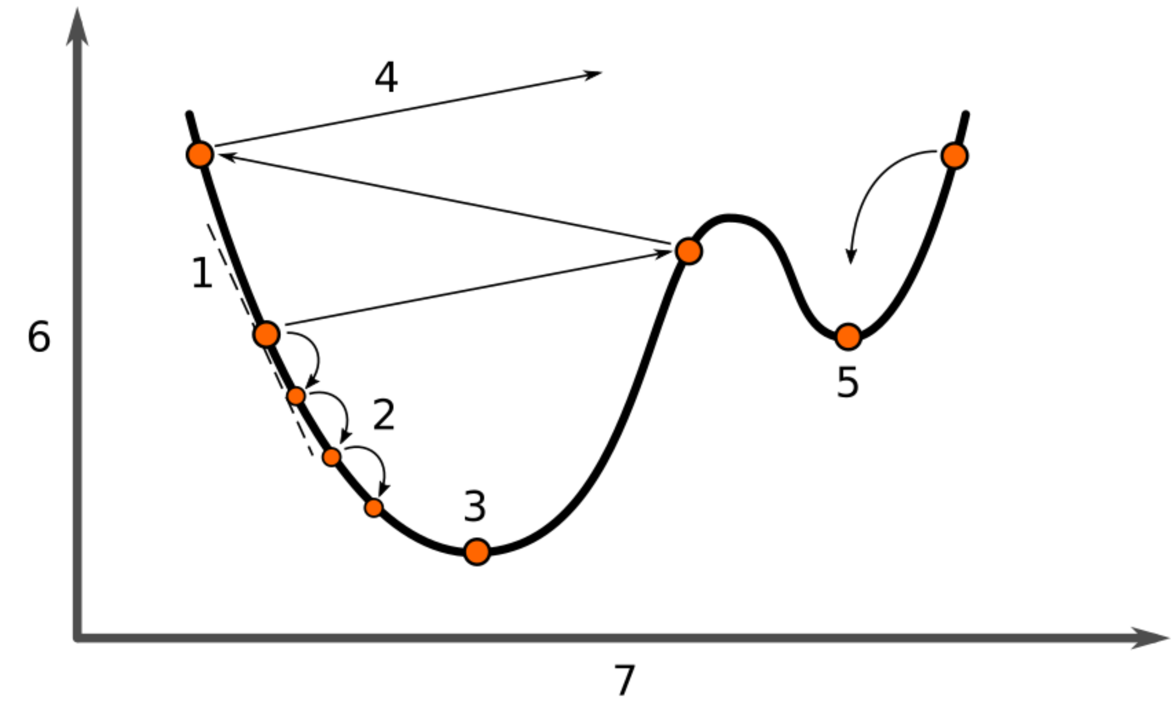
Weight Initialization Basics

- When **Randomly** initialize the weights
  - There is a chance to have bad start points
  - **Vanishing Gradients**
  - **Exploding Gradients**
  - **Symmetry Breaking**
  - **Unstable Learning Dynamics**



Impact of Weight initialization

- Training Speed
- Model Accuracy
- Prevention of Overfitting



Common Initialization Strategies

- **Random Initialization**

$$\omega \sim Uniform(-\epsilon, \epsilon)$$

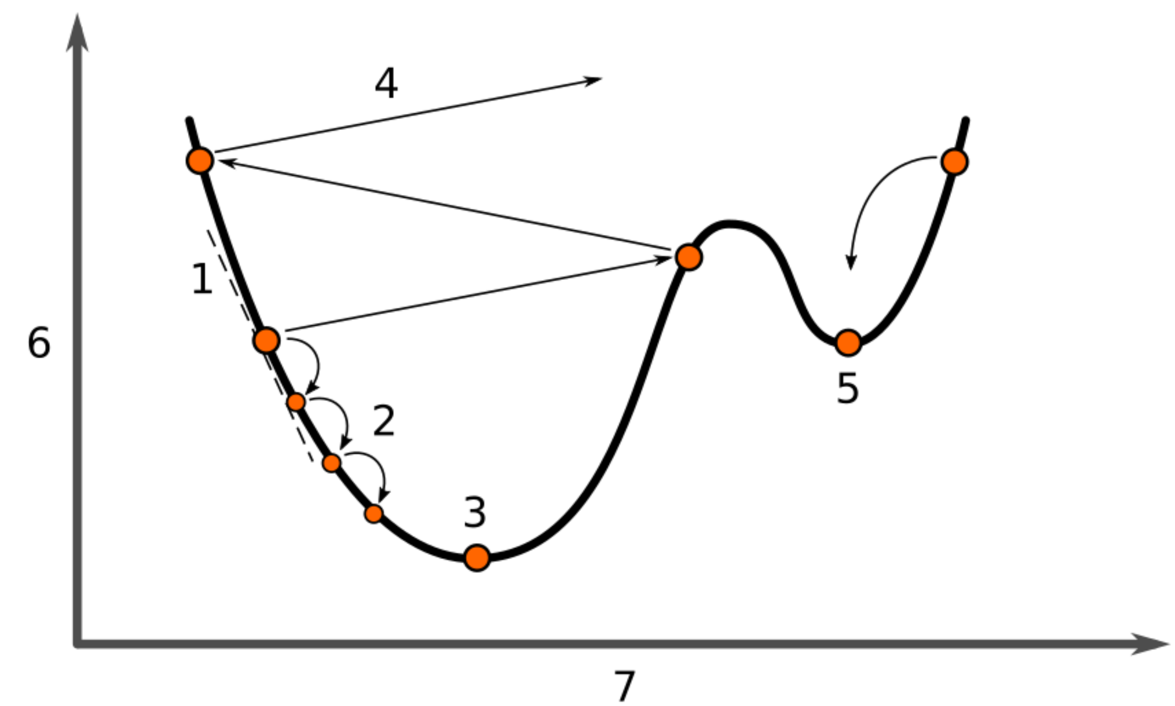
- **Xavier/Glorot Initialization**

$$\omega \sim Uniform(-\sqrt{\frac{1}{n}}, \sqrt{\frac{1}{n}})$$

- **He Initialization**

$$\omega \sim \mathcal{N}(0, \sqrt{\frac{2}{n}}) \textbf{ or } \omega \sim U(-\sqrt{\frac{6}{n}}, \sqrt{\frac{6}{n}})$$

- **Zero Initialization**
  - Should be avoided for weights but can be used for bias initialization



Vanishing Gradient Problem

- It happens when gradients become extremely small during backpropagation
- **Root Causes**
  - **Activation Functions**
    - Saturating functions like **sigmoid** and **Tanh**
      - In case of Sigmoid: $f'(x) = f(x)(1 - f(x))$, becomes very small if $f(x)$ is close to 0 or 1
  - **Weight Initialization**
  - **Deep Networks**
  - **Loss Function**
    - Functions with **small derivatives** like the ones used in **classification tasks**

Vanishing Gradient Problem

- **Impacts on Deep Network**
  - Slow Learning in early layers
  - Poor model performance
  - Optimization difficulties
  - Bias toward output layers

How To Identify Vanishing Gradient

- **Gradient Magnitude Inspection**
  - Measure the magnitude of gradients at different layers during training
    - The early layers will have near-zero gradient values
- **Training Behavior**
  - Slow Convergence
- **Activation Distribution**
  - Saturation near min/max of the activation range, suggests potential gradient vanishing
- **Weight Updates**
  - Small changes in weights of earlier layers during training

МФТИ

How To Mitigate Vanishing Gradient

- Use NonSaturating Functions like ReLU
- Weight Initialization Techniques
- Batch Normalization
- Residual Networks?!

Exploding Gradient Problem

- It happens when gradients grow uncontrollably large during backpropagation
- **Root Causes**
  - **Deep Networks**
    - If gradients in different layers are larger than 1, then the repeated multiplication of them would be huge
  - **Recurrent Neural Networks (RNNs)**
    - Gradients are propagated through time and long sequences amplify the effect
  - **Improper weight Initialization**
  - **Activation Functions with unbounded outputs**

How To Identify Exploding Gradient

- **Gradient Magnitude Inspection**
  - Measure the magnitude of gradients at different layers during training
- **Training Behavior**
  - Observe instabilities, Diverging loss values, NaN values
- **Activation Distribution**
- **Log Gradient Values**
  - Visualize gradients using **TensorBoard.** Even a sudden spike is a sign.

МФТИ

How To Mitigate Vanishing Gradient

- **Gradient Clipping**

$$If \|\nabla L\| > \tau,\ \nabla L \leftarrow \tau \,.\, \frac{\nabla L}{\|\nabla L\|}$$

- **Weight Regularization**
  - **L2** regularization penalizes the large weights
- **Proper Weight Initialization**
  - **Xavier/Glorot** or **He** initialization ensure the weights are scaled properly
- **Optimizers with adaptive learning**
  - **Adam** or **RMSProp**, dynamically adjut the learning rate for each parameter
- **Batch Normalization**
- **Architectural Adjustments**
  - For **RNNs,** you might use **LSTMs** or **GRUs** which include gating mechanisms to control the gradient flow

МФТИ

Initialization Techniques

- **Zero Initialization**
  - **Pros:** Simple, and works for biases
  - **Cons:** All neurons in a layer learn the same features because their gradients are identical
- **Random Initialization**
  - **Pros:** Breaking Symmetry and Enables effective learning
  - **Cons**: Randomly chosen values can result in vanishing/exploding gradients
- **He Initialization**
  - **Pros:** Prevents exploding/vanishing gradients
  - **Cons**: Might not be optimal for activations **other than ReLU,** and a slightly more computation overhead

Initialization Techniques

Method	Pros	Cons	Best for
Zero Initialization	Simple	Symmetry	Bias Initialization
Random Initialization	Breaks Symmetry	May cause vanishing/ exploding gradients	Shallow networks
He Initialization	Prevents gradient issues	Slight computational overhead	Deep networks with ReLU