**📘 Gradient Descent – Study Notes**

**🔹 What is Gradient Descent?**

Gradient Descent is an **optimization algorithm** used to minimize a **loss function** by iteratively updating the model's parameters in the direction of the steepest descent (negative gradient).

**❗ Formula:**

θ =θ−α⋅∇J(θ)

* θ theta: Parameters (weights)
* α alpha: Learning rate
* ∇J(θ): Gradient of the loss function

**🔸 Types of Gradient Descent**

| **Type** | **Description** |
| --- | --- |
| **Batch Gradient Descent** | Uses the **entire dataset** to compute the gradient each step. |
| **Stochastic GD (SGD)** | Updates parameters using **one random data point** at a time. |
| **Mini-Batch GD** | Uses **small batches (e.g., 32, 64)** of data to update parameters. |

**🔸 Advanced Variants (Optimizers)**

| **Optimizer** | **Key Idea** |
| --- | --- |
| **Momentum** | Adds a fraction of the previous update to smooth updates. |
| **AdaGrad** | Adapts learning rate for each parameter based on frequency. |
| **RMSProp** | Uses moving average of squared gradients to scale updates. |
| **Adam** | Combines Momentum and RMSProp for better performance. |

**Pros and Cons Table**

| **Type** | **Pros** | **Cons** |
| --- | --- | --- |
| **Batch GD** | - Stable - Accurate gradients | - Slow for large data - High memory use |
| **Stochastic GD (SGD)** | - Fast  - Good for online learning - Escapes local minima | - Noisy  - May fluctuate - Needs learning rate tuning |
| **Mini-Batch GD** | - Efficient - Stable updates - GPU-friendly | - Needs batch size tuning - May still get stuck |
| **Adam Optimizer** | - Adaptive learning rate - Fast convergence | - More memory - Many hyperparameters |
| **AdaGrad** | - Great for sparse data | - Learning rate can decay too much |
| **RMSProp** | - Works well with RNNs | - Sensitive to tuning parameters |

**🔸 Summary**

* Use **SGD** for fast, online learning.
* Use **Mini-Batch GD** for balanced performance.
* Use **Adam** for deep learning and complex models.
* Tune **learning rate (α)** and **batch size** for best results.

**🧠 When to Use Gradient Descent in Regression Models**

**✅ Use Gradient Descent When:**

| **Scenario** | **Explanation** |
| --- | --- |
| 🔸 **Large datasets** | For huge datasets, computing closed-form solutions (like the normal equation in linear regression) is expensive and slow. Gradient Descent is more efficient and scalable. |
| 🔸 **Multiple features (high-dimensional data)** | Analytical methods become computationally heavy with many features. Gradient descent handles this efficiently. |
| 🔸 **Custom or complex loss functions** | If you're using a loss function that doesn't have a simple derivative (e.g., in robust regression), gradient descent is necessary. |
| 🔸 **Regularized models** | Models like Lasso (L1) or Ridge (L2) regression often use gradient-based optimizers because closed-form solutions are more complex or unavailable. |
| 🔸 **Online/streaming learning** | With data arriving in streams, **Stochastic Gradient Descent (SGD)** allows real-time model updates without storing all data. |
| 🔸 **Neural networks or non-linear regression** | For deep learning or non-linear models, there's no analytical solution—gradient descent is the standard approach. |