**🎓 Lecture: Learning Rate and Learning Rate Schedules**

**1. Parameters vs. Hyperparameters**

* **Parameters** = weights and biases of the model (learned during training).
* **Hyperparameters** = settings we choose *before* training. Examples:
  + Number of hidden units (width)
  + Number of hidden layers (depth)
  + Learning rate ()
* Hyperparameters are **not learned automatically**; we must pick and tune them.

**2. The Learning Rate ()**

The learning rate is one of the **most important hyperparameters**.

* If is **too large** → model oscillates around the minimum or diverges (blows up).
* If is **too small** → training is very slow (takes forever to converge).
* So: it must be **big enough for speed**, but **small enough for stability**.

👉 Example: In earlier exercises (Excel demo, linear model), you could play with learning rate and see this effect.

**3. Learning Rate Schedules (Smarter Approach)**

Instead of keeping constant, we can **change it during training**.  
This way we get the best of both worlds:

* Start **high** → faster progress at the beginning.
* End **low** → stable, accurate convergence.

**🔹 (a) Piecewise Constant Schedule**

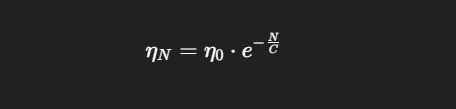
* Divide training into stages.
* Example:
  + Epochs 1–5 →
  + Epochs 6–10 →
  + Epochs 11–end →

Pros: Simple, works better than constant learning rate.  
Cons: Crude → requires guessing how many epochs training will need.

**🔹 (b) Exponential Schedule**

A smoother and smarter alternative.

Formula:



Where:

* = initial learning rate (e.g., 0.1)
* = current epoch number
* = constant (controls decay speed)

👉 Example: If , the learning rate shrinks smoothly every epoch.

How to pick ?

* Roughly the same order of magnitude as the number of epochs you expect.
* If training takes 100 epochs → between 50 and 500 is fine.
* If training takes 1000 epochs → between 500 and 5000 works.
* In practice, exact value of is **less important** → just using a schedule makes a big difference.

**4. Trade-Off: More Hyperparameters**

Adding **momentum** and **learning rate schedules** makes optimization much better.  
But… each introduces **extra hyperparameters** ( for momentum, for schedule).

* Usually, rule-of-thumb values (like , ) work well.
* But for specific problems, tuning may be needed.
* Always worth testing a few values before deciding.

✅ **Summary**

* Learning rate is a **key hyperparameter**.
* Must balance speed and stability.
* **Learning rate schedules** solve this by starting high, then decaying.
* Two main types:
  + **Piecewise constant** (simple, but crude)
  + **Exponential schedule** (smooth, practical, widely used)
* Adds more hyperparameters, but improves results significantly.

**🌱 The Relationships**

**1. Training Loss**

* This is the **error on the training data** (how badly the model is performing while learning).
* Goal: Make training loss go **down** over time.

**2. Validation Loss**

* This is the **error on unseen data** (not used for training).
* Tells us if the model is actually **generalizing**.
* If training loss ↓ but validation loss ↑ → the model is **overfitting** (memorizing training data instead of learning patterns).

**3. Gradient Descent**

* Method we use to **reduce training loss**.
* Works by updating weights and biases step by step:
* Without gradient descent, the model wouldn’t know how to adjust itself.

**4. Backpropagation**

* The **mechanism** that tells gradient descent how to update weights.
* Backpropagation computes the gradients () using the chain rule.
* In short:
  + Forward pass → compute predictions + loss.
  + Backward pass → compute gradients.
  + Gradient Descent → update weights with those gradients.

👉 Backprop = *how we calculate gradients*.  
👉 Gradient Descent = *how we use those gradients to update*.

**5. Exponential Learning Rate Schedule**

* A trick to make **gradient descent work better**.
* Starts with a **big learning rate** (fast progress), then shrinks it exponentially (precise fine-tuning).
* Prevents overshooting and makes both training loss and validation loss decrease more smoothly.

**🔗 How They All Connect (Simple Story)**

1. **Backpropagation** calculates the gradients of the loss function.
2. **Gradient Descent** uses those gradients (and the learning rate) to update weights and biases.
3. With each update, the **training loss** decreases.
4. If everything goes well, the **validation loss** also decreases → model generalizes well.
5. To make this process efficient and stable, we use an **Exponential Learning Rate Schedule** → fast learning at the start, careful learning at the end.

✅ **Simplified Analogy:**

* Think of training a model like **learning to park a car**.
* **Training loss** = how far you are from the parking spot while practicing.
* **Validation loss** = how well you park when someone gives you a *new* car (unseen case).
* **Backpropagation** = your brain figuring out “turn left/right” based on mistakes.
* **Gradient Descent** = you actually turning the wheel step by step.
* **Exponential Learning Rate Schedule** = starting fast to reach the spot, then slowing down for precise parking.