✅ Tóm tắt nhanh:

- Giai đoạn 1 (hoàn thành):
  - Đã thêm **noise** vào mô hình GP ⇒ Không cần thêm bước này nữa ở giai đoạn 2.
  - Đã phát triển cơ bản 1 kernel (likely RBF).
- Quyết định ở Giai đoạn 2:
  - **Không phát triển thêm nhiều loại kernel** (ví dụ Matern, RationalQuadratic... ⇒ bỏ qua để tiết kiệm thời gian).
  - **Có thể** sẽ nghiên cứu thêm việc **kết hợp kernels** (kernel combination).
  - **Mục tiêu chính**: **Tối ưu hyper-parameters**.

🎯 Kế hoạch cho Giai đoạn 2:

1. **Tối ưu hoá Hyperparameters**:
   - Các hyperparameters cần tối ưu có thể là:
     - Lengthscale (của RBF kernel hoặc kernels khác nếu có)
     - Variance (outputscale)
     - Noise variance
   - Sử dụng:
     - **Gradient Descent** (thông qua tối ưu hoá marginal log likelihood - MLL)
     - **Adam optimizer** (nếu muốn thử cách học tiên tiến hơn).
2. **Tối ưu kernel combination** (nếu quyết định triển khai thêm):
   - Học cách kết hợp kernels như: `RBF + Linear`, `RBF * Periodic`, etc.
   - Xây dựng class KernelCombination nếu cần.

📂 Cập nhật thêm vào file cấu trúc:

Trong cấu trúc project của chúng ta (theo Readme cũ), Giai đoạn 2 sẽ thêm các file/hàm như sau:

File	Nội dung mới cần thêm
`optimization/hyperparameter_optimization.py`	Hàm tối ưu hóa hyperparameters bằng việc maximize Marginal Log Likelihood (MLL)
`models/kernels/combination_kernel.py`	(Tùy chọn) Xây dựng class cho kernel combination nếu cần
`trainers/train_gp.py`	Gọi quá trình training + tối ưu hyperparameters

📌 Hỏi nhanh bạn trước khi mình bắt tay vào code:

1. **Bạn có muốn tối ưu hyperparameters theo cách thủ công** (self-train optimizer + loss) hay **sử dụng thư viện hỗ trợ** như GPyTorch Optimizer?

2. **Bạn có muốn mình viết thêm luôn lựa chọn kernel combination** ngay trong giai đoạn 2, hay để riêng thành Giai đoạn 3?

Nếu bạn confirm (OK) hoặc chọn một số option, mình sẽ tiến hành **triển khai code** theo đó luôn nha 🚀

Bạn trả lời nhanh thế này nhé:

- "Tối ưu tự viết" / "Dùng thư viện"
- "Kernel Combination: Có" / "Kernel Combination: Không"

Rõ ràng rồi! ✍️
Bạn chọn:

- **Tối ưu tự viết** (chúng ta sẽ tự code hàm tối ưu hyperparameters, không dùng thư viện ngoài).
- **Không làm Kernel Combination** (chỉ tập trung tối ưu single kernel đã có thôi).

🎯 Kế hoạch chi tiết mình sẽ triển khai ngay sau đây:

✅ Viết file mới:
`optimization/hyperparameter_optimization.py`

✅ Các phần chính trong file này:

1. **Tính Marginal Log Likelihood (MLL)**:

$$\log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\mathbf{y}^T K^{-1}\mathbf{y} - \frac{1}{2}\log|K| - \frac{n}{2}\log(2\pi)$$

   - $$K = K(X, X) + \sigma_n^2 I$$

2. **Gradient Descent tối ưu hyperparameters**:
   - Tối ưu hóa các tham số:
     - Lengthscale
     - Variance
     - Noise variance
   - Sử dụng công thức đạo hàm (hoặc numerical approximation nếu cần).

3. **Hàm huấn luyện**:
   - Tự cập nhật hyperparameters sau mỗi epoch.

📦 Sau khi mình code xong, bạn sẽ có:

- Hàm `train_hyperparameters(gp_model, X_train, y_train, epochs=100, lr=0.01)`
- Gọi `gp_model` để tự học các hyperparameters mới ⇒ Sau đó có thể dự đoán với các tham số tối ưu.

Mình sẽ **bắt tay viết file** ngay bây giờ theo đúng phong cách clean-code để bạn vừa dễ đọc vừa dễ học nhé 🔥 .

Bắt đầu nào 🚀 :

✅ **Xong phần tối ưu tự viết** rồi nha!
Tóm gọn lại:

- Dùng **Finite Difference** để tính gần đúng gradient.
- Dùng **Gradient Ascent** để tối đa hóa Marginal Log Likelihood.
- Code rất gọn, chỉ cần `optimizer.optimize(X_train, y_train, epochs=100)` là tự động tối ưu.

Bây giờ bạn muốn mình làm thêm bước nào nữa không?
Ví dụ như:

- Viết thêm **visualize** biểu đồ MLL theo epochs 📈
- **Gọi thử** 1 ví dụ tối ưu hyperparameters thực tế 🚀
- Hoặc **kết nối** file tối ưu này vào GP model của bạn cho pipeline tự động hơn.

Bạn chọn nhé! 🔥 💬