Tuyệt vời! 🎯
Cảm ơn bạn đã thống nhất về cách định dạng toán nhé — mình sẽ tiếp tục dùng kiểu quen của mình là:

- (\text{inline math: } \LaTeX) với `\(...\)`
- [\text{block math: } \LaTeX] với `\[...\]`

✅ Bây giờ, chúng ta chính thức **bắt đầu Giai đoạn 4.1: Independent Multi-output GP Classification**!

📚 Bố cục hướng dẫn Giai đoạn 4.1

Mình sẽ bám đúng phong cách mấy file trước:

Phần	Nội dung
1	Hiểu bản chất Multi-output GP Classification
2	Công thức tổng quát
3	Sơ đồ luồng (Flowchart)
4	Pseudo-code cấp cao
5	Dự kiến cấu trúc thư mục/code
6	Checklist nhỏ cho Giai đoạn 4.1

🧠 1. Hiểu bản chất: Independent Multi-output GP Classification

---

❓ Bài toán đặt ra:

- Input: (\mathbf{X} \in \mathbb{R}^{n \times d}) (n samples, d features).
- Output: (\mathbf{Y} \in {0,1}^{n \times T}) (n samples, T labels).

👉 Tức là mỗi input (\mathbf{x}_i) có **T** nhãn nhị phân tương ứng.

❗ Ý tưởng then chốt:

- Với mỗi label (t \in {1, \dots, T}):
  - Huấn luyện **một mô hình GP Classification độc lập**.
- Sau đó khi dự đoán:
  - Dự đoán xác suất từng label riêng rẽ.
  - Gom thành vector output.

📋 So sánh nhanh:

GP Regression	GP Classification (binary)	Multi-output GP Classification
Predict 1 real value	Predict 1 label (0/1)	Predict T labels (0/1)
Gaussian likelihood	Bernoulli likelihood	Bernoulli likelihood cho mỗi task
Single GP model	Single GP model	T GP models

📐 2. Công thức tổng quát

Latent function cho task (t):

[
f^{(t)}(x) \sim \mathcal{GP}(0, k(x, x'))
]

(kernel có thể shared hoặc riêng — ở đây ta dùng **shared kernel** cho đơn giản.)

Likelihood cho mỗi output:

[
p(y^{(t)}_i = 1 | f^{(t)}(x_i)) = \sigma(f^{(t)}(x_i))
]
trong đó (\sigma(\cdot)) là hàm sigmoid hoặc probit.

Posterior xấp xỉ cho mỗi task (Laplace Approximation):

Sau khi tối ưu:

[
p(f^{(t)} | X, y^{(t)}) \approx \mathcal{N}(\hat{f}^{(t)}, \Sigma^{(t)})
]
với:

- (\hat{f}^{(t)}): Mode của posterior cho task (t).
- (\Sigma^{(t)} = (K^{-1} + W^{(t)})^{-1}).

(W^{(t)}) là Hessian matrix của negative log-likelihood cho task (t).

Predictive distribution:

Với một điểm test (x_*):

Predictive mean latent function:

$$
\mu_^{(t)} = k_^\top (K + W^{(t)-1})^{-1} \hat{f}^{(t)}
$$

Predictive variance latent function:

$$
\sigma_^{2(t)} = k(x_, x_) - k_^\top (K + W^{(t)-1})^{-1} k_*
$$

Chuyển sang xác suất phân loại:

Ví dụ với sigmoid link:

$$
p(y_^{(t)} = 1 | x_) = \sigma\left(\frac{\mu_^{(t)}}{ \sqrt{1 + \frac{\pi}{8} \sigma_^{2(t)} } } \right)
$$

(Công thức này sử dụng xấp xỉ logistic function khi predict.)

🔥 3. Flowchart Giai đoạn 4.1

```
Start
  ↓
Input (X, Y multi-label)
  ↓
For each output (task t):
  ↓
  Create Single GP Classifier (with shared kernel)
  ↓
  Train (Laplace Approximation)
  ↓
  Store model parameters
  ↓
Predict:
  ↓
For each output (task t):
  ↓
  Predict latent mean, variance
  ↓
  Compute probability
  ↓
Stack all outputs
  ↓
Output (multi-label prediction)
```

🛠️ 4. Pseudo-code cấp cao (Full Pipeline)

```
class SingleTaskGPClassifier:
    def __init__(self, kernel):
        ...

    def train(self, X, y, epochs=100):
        # Laplace Approximation Training
        ...

    def predict(self, X_test):
        # Predict latent mean and variance
        # Then compute probability
        return probas  # shape (n_test, )

class MultiOutputGPClassifier:
    def __init__(self, num_outputs, kernel):
        self.models = [SingleTaskGPClassifier(kernel) for _ in
range(num_outputs)]

    def train(self, X, Y, epochs=100):
        for t in range(self.num_outputs):
            self.models[t].train(X, Y[:, t], epochs)

    def predict(self, X_test):
        outputs = []
        for t in range(self.num_outputs):
            outputs.append(self.models[t].predict(X_test))
        return np.stack(outputs, axis=1)  # shape (n_test, num_outputs)
```

📂 5. Dự kiến cấu trúc code

Folder/File	Nội dung
models/single_task_gp_classifier.py	Class SingleTaskGPClassifier
models/multioutput_gp_classifier.py	Class MultiOutputGPClassifier
trainers/train_multioutput_gp.py	Hàm train MultiOutputGPClassifier
predictors/predict_multioutput_gp.py	Hàm predict MultiOutputGPClassifier
utils/kernels.py	Kernel function (RBF Kernel)

✅ 6. Checklist nhỏ cho Giai đoạn 4.1

STT	Việc cần làm	Trạng thái
1	Viết SingleTaskGPClassifier	⬜
2	Viết MultiOutputGPClassifier	⬜
3	Viết trainer cho MultiOutput	⬜
4	Viết predictor cho MultiOutput	⬜
5	Test trên toy multi-label dataset	⬜

🚀 Checkpoint 2:

Bạn xác nhận nhé:

- ✅ Flow tổng thể đã rõ ràng?
- ✅ Bắt đầu từ việc mình viết `SingleTaskGPClassifier` trước, rồi build `MultiOutputGPClassifier` dựa trên nó?

👉 Nếu bạn OK, mình sẽ lập tức **bắt tay vào viết file `models/single_task_gp_classifier.py` đầu tiên** theo đúng phong cách dễ đọc như các phase trước.

💬 Bạn trả lời nhanh:

- **"OK, viết SingleTaskGPClassifier trước"**
- hoặc **"Khoan, mình có câu hỏi về kernel sharing / khác biệt outputs"**

Chờ lệnh của bạn! 🚀
(Đang sẵn sàng code sạch siêu chi tiết luôn!)