KNOWLEDGE RISE

PROCESS PARAMETER OPTIMIZATION IN ADDITIVE MANUFACTURING

Lecture 2 of 3

Dr. Nazarii Mediukh,
Institute for Problems of Materials Science, NASU

Supported by

EIT HEI Initiative eit-hei.eu

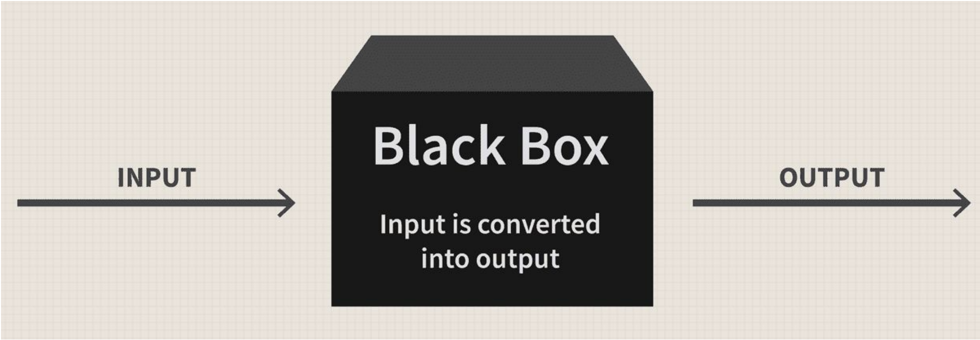
CloudEARTHi.com

eit RawMaterials Connecting matters

Co-funded by the European Union

Recap

What did we cover?
- ML fundamentals: supervised, unsupervised, reinforcement learning
- Data challenges in AM: small samples, noise, imbalance
- The ML pipeline: data → preprocessing → model → validation



ML helps us to learn patterns from data, but now we ask:
“HOW DO WE FIND THE BEST PARAMETERS?”

Traditional approach: trial and error
- Test 5 params × 10 values each = 50 - 100000 experiments
- Misses parameter interactions
- Time-consuming and expensive

ML approach: Smart exploration
- Learn from fewer experiments
- Model parameter interactions
- Quantify uncertainty
- Guide next experiments intelligently

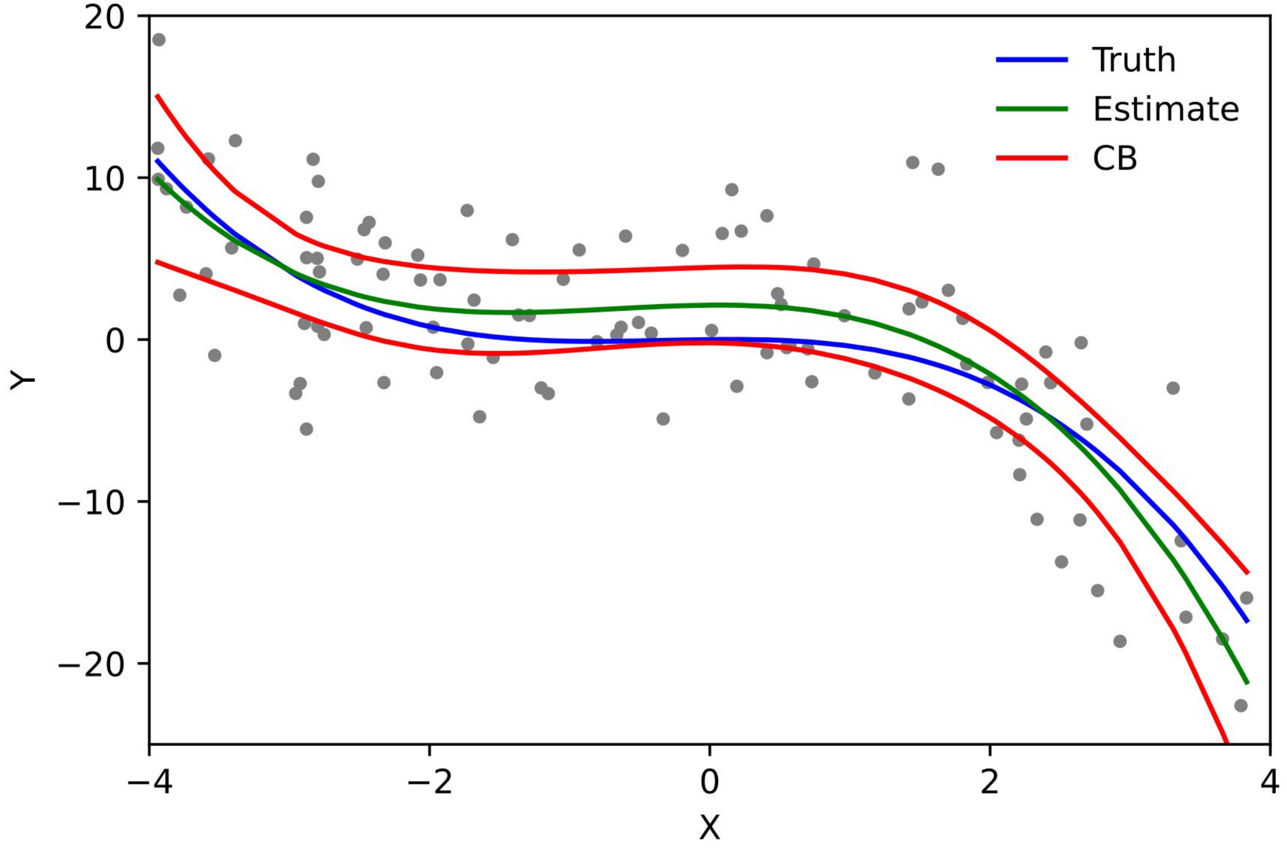
Course roadmap

Lecture 1: Foundations
- ML basics
- Data in AM
- The ML pipeline

Lecture 2: Process optimization (Today)
- Regression techniques
- Bayesian optimization
- Multi-fidelity approaches

Lecture 3: Advanced Applications
- Multi-objective optimization
- Deep learning for defects
- Real-time process control

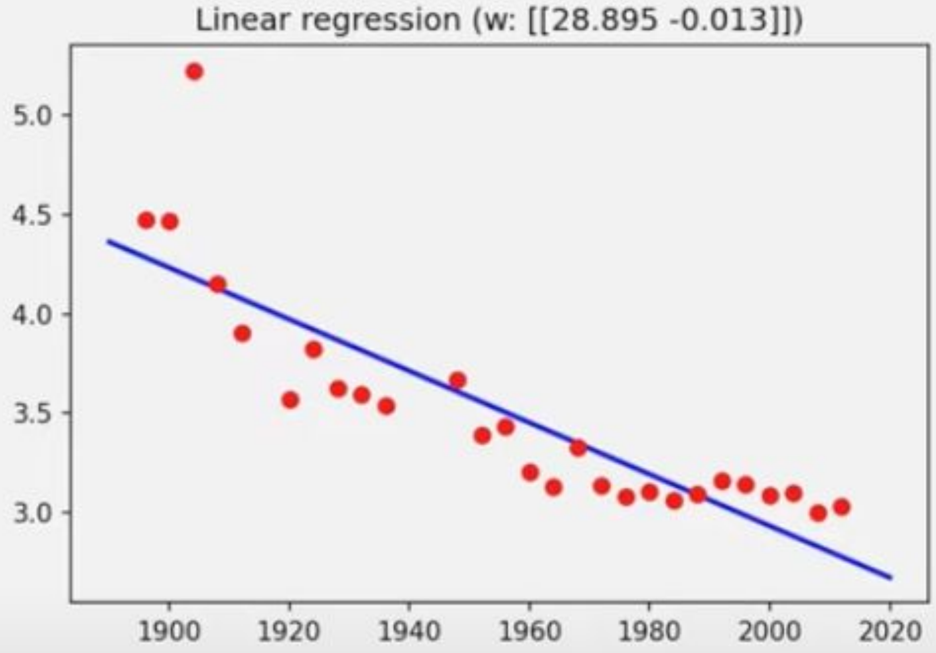
What is regression?



Linear regression

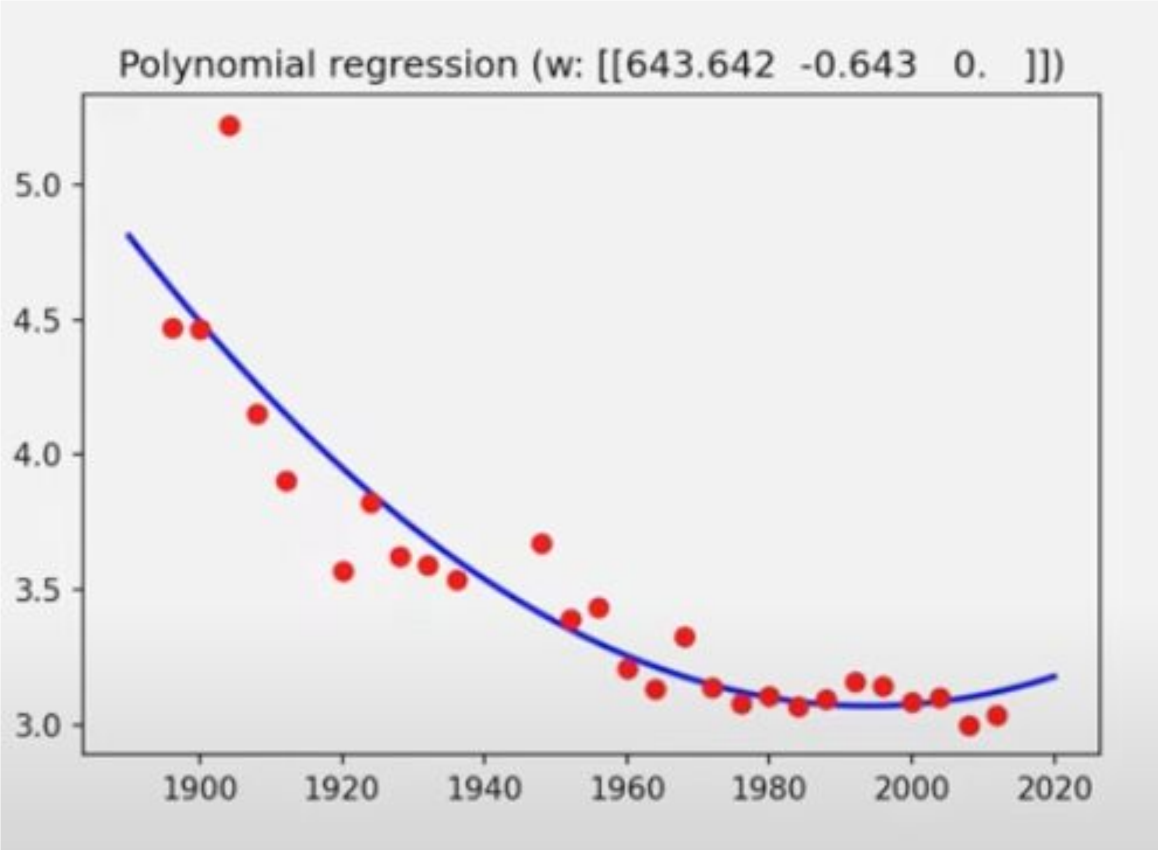
$$y = f(\mathbf{x}_i) = \mathbf{x}_i \mathbf{w} + b$$

We learned: $y = w_1 x + w_0 = -0.013x + 28.895$

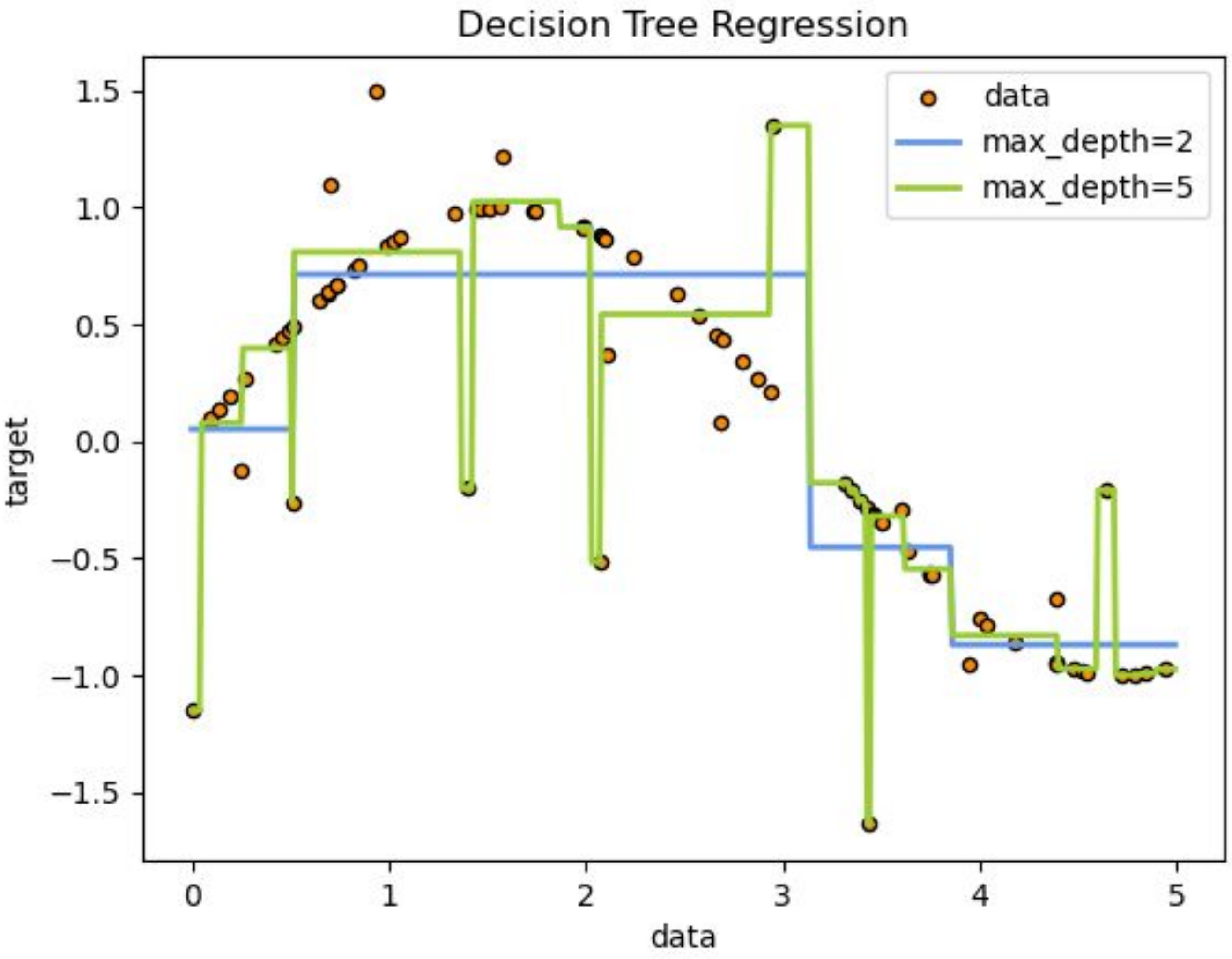


Polinomial regreession

$$\Phi = \begin{bmatrix} 1 & \mathbf{x} & \mathbf{x}^2 \end{bmatrix}$$



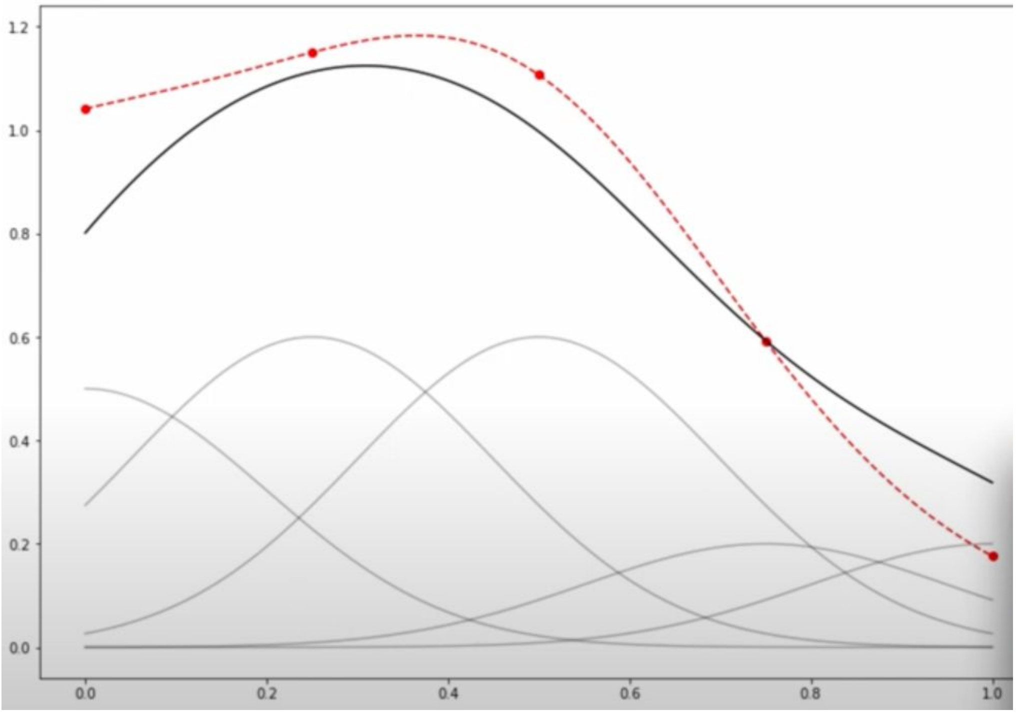
Decision tree regression

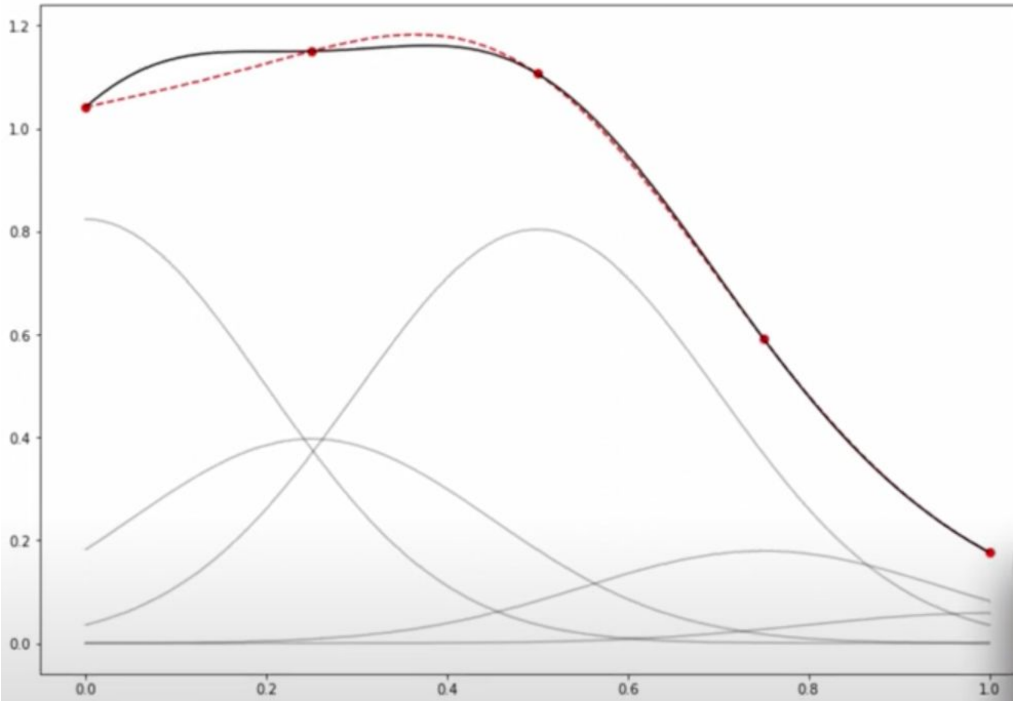


Kerner ridge regression

$$f : \mathcal{X} \rightarrow \mathcal{Y},$$

$$\hat{f}(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x).$$

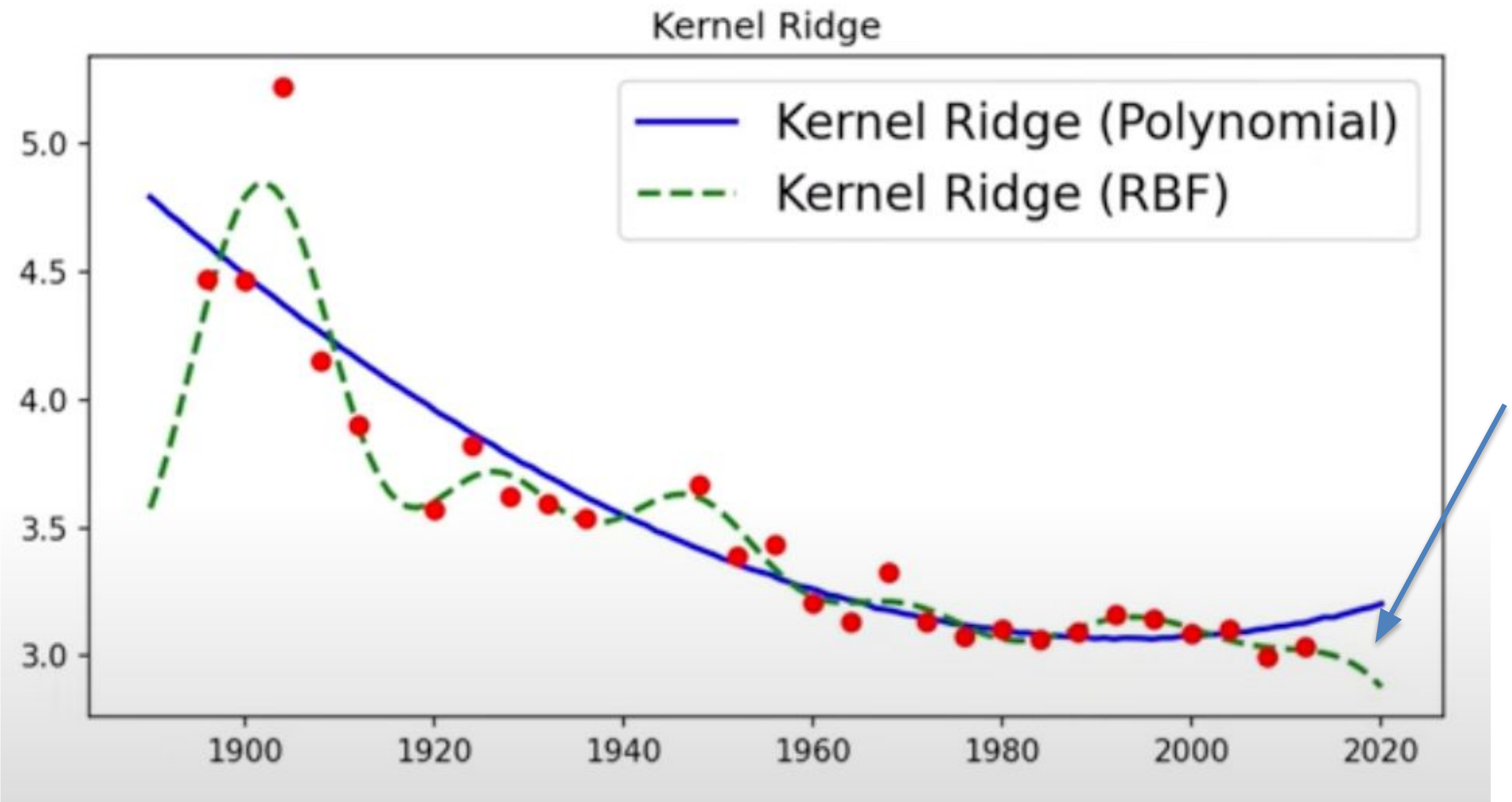




Kerner ridge regression

Radial basis function (RBF) kernel

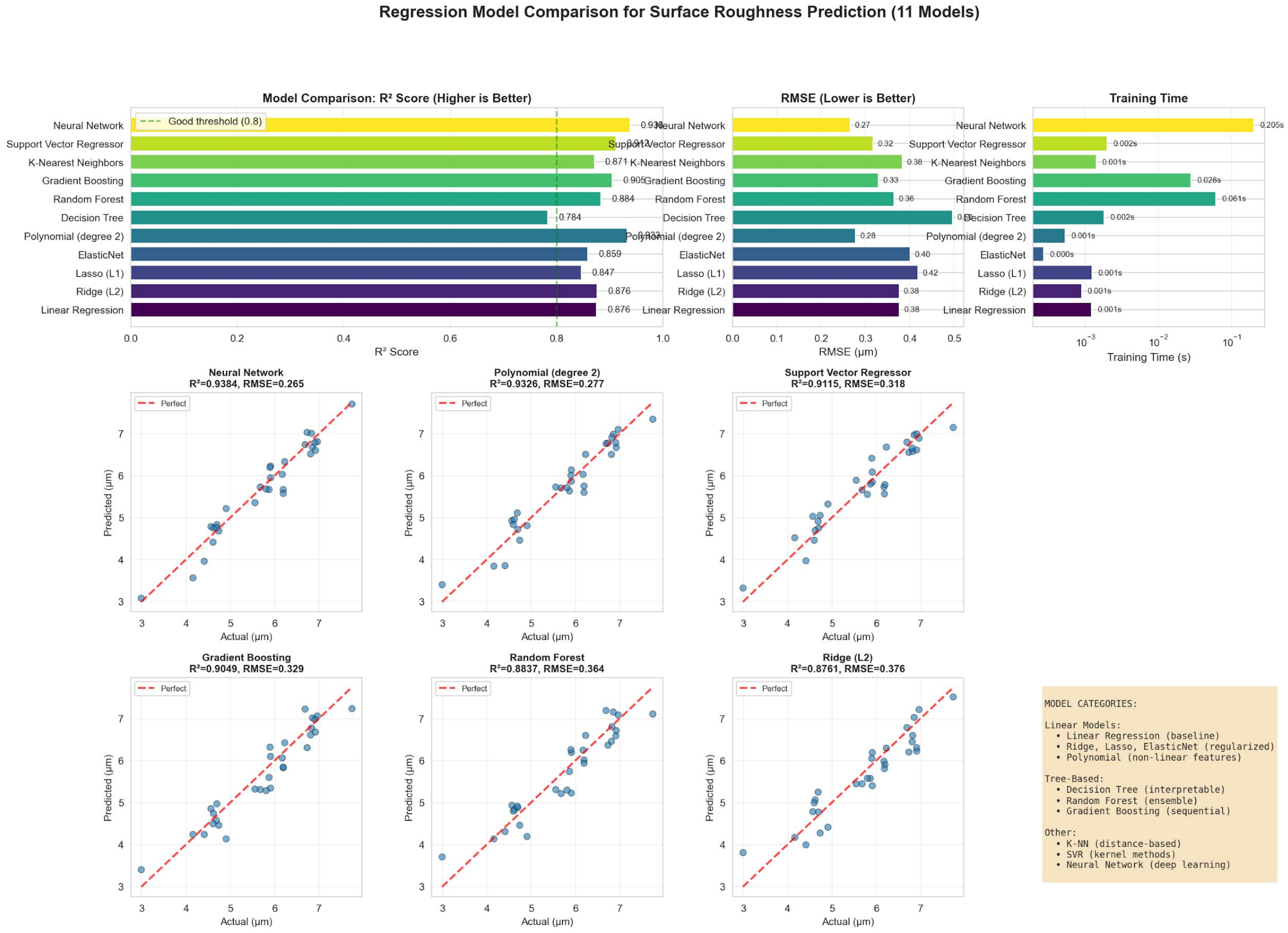
$$k(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j)^2}{2l^2}\right)$$



Compare regression methods

Algorithm	Strengths	Weaknesses	Best For
Linear Regression	Fast, interpretable, stable	Only linear relationships	Initial exploration, simple processes
Polynomial Regression	Captures non-linearity	Overfits easily	Known polynomial relationships
Random Forest	Handles non-linearity, robust	Black box, no uncertainty	Medium data (100-1000 samples)
Gradient Boosting	High accuracy, feature importance	Slower, needs tuning	Prediction accuracy critical
SVR	Good with small data	Hard to tune, slow	Limited data (20-100 samples)
Gaussian Process	Uncertainty quantification!	Slow with large data	Optimization, small data

Demo 1: compare regression methods

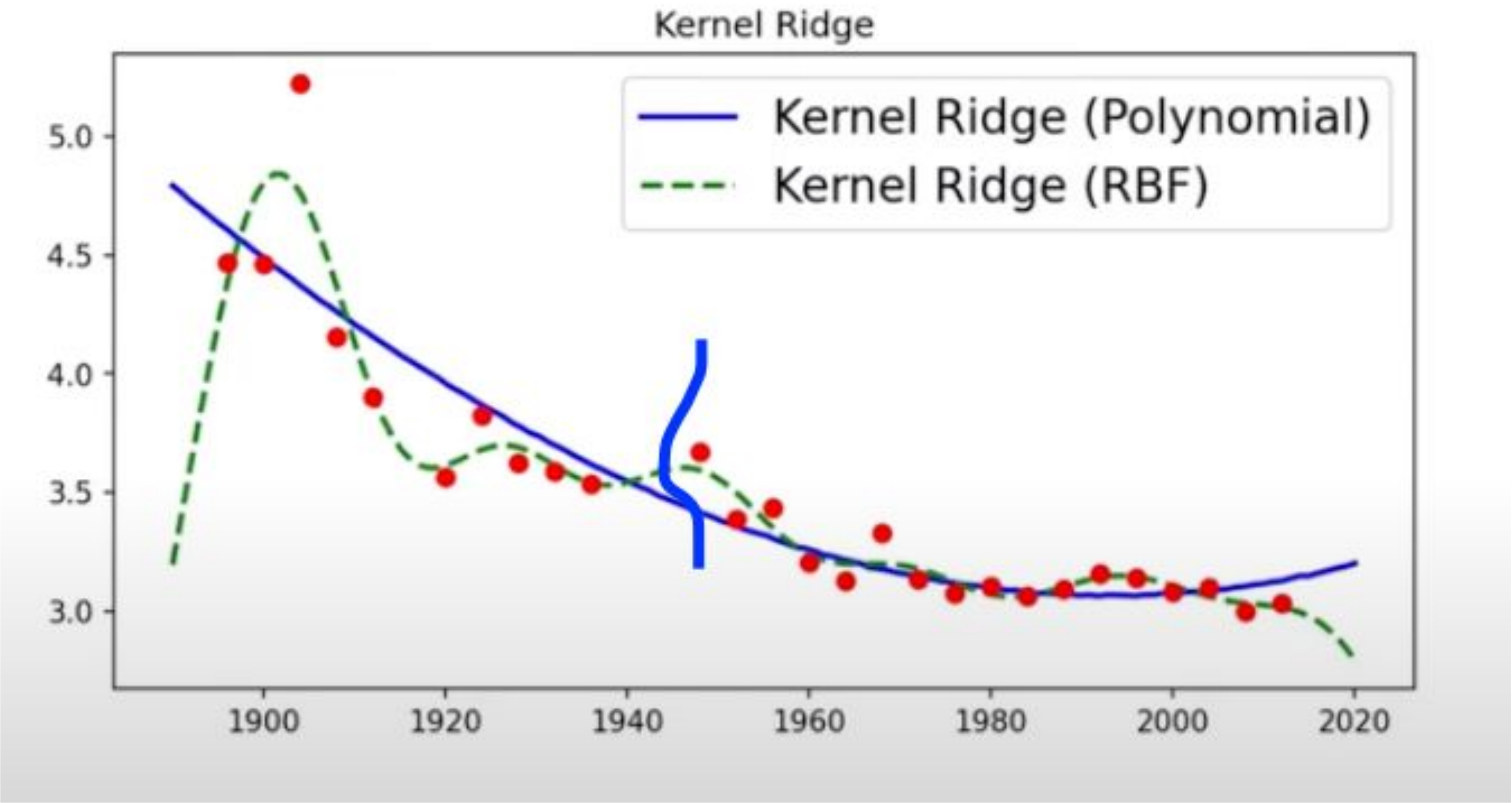


Probabilistic interpretation of regression

$$y_i = w_1 x_i + w_0 + \epsilon_i.$$

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

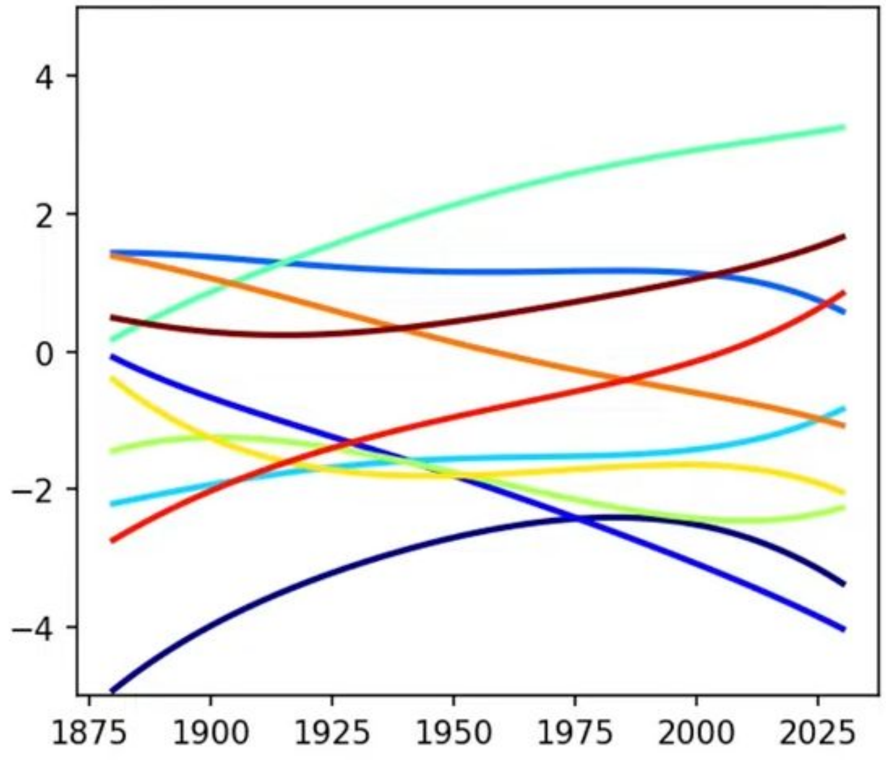
$$y = \mathcal{N}(\mathbf{w}\mathbf{x}, \sigma^2)$$

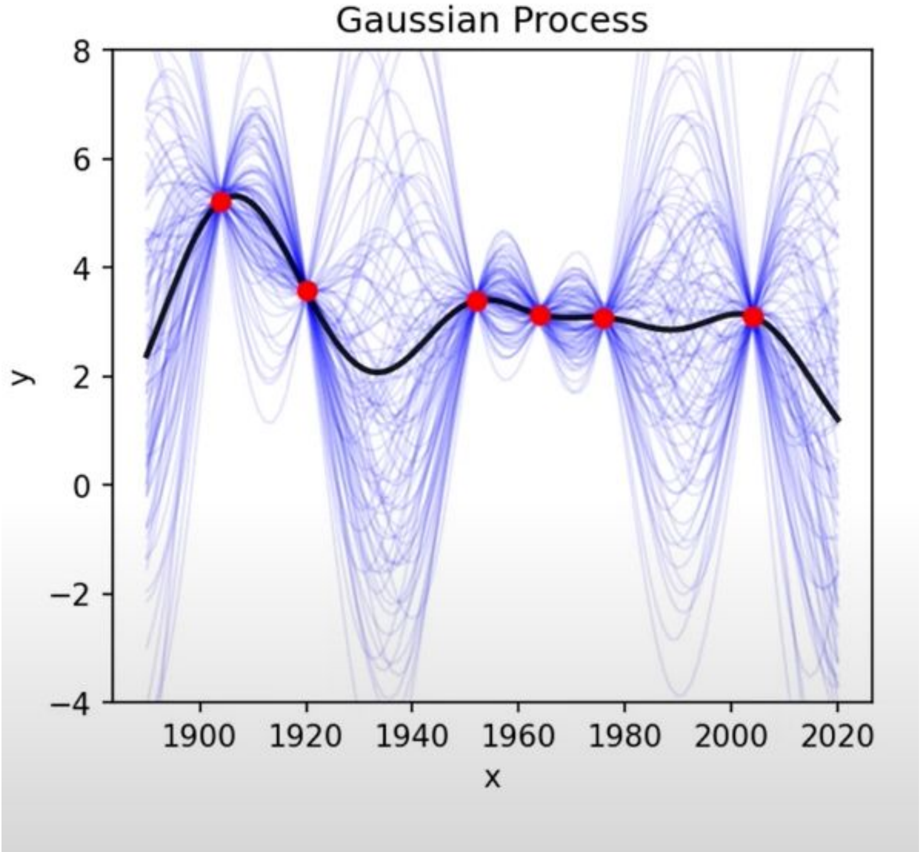


Ways to learn on the data with noise

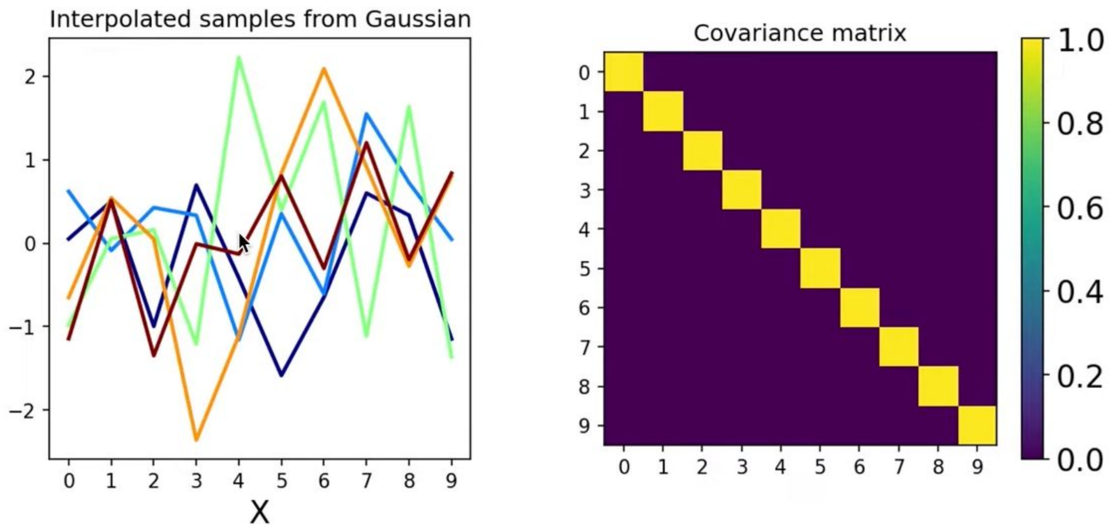
Maximum Likelihood Estimation (MLE):
Learn w so P(X|w) is maximised

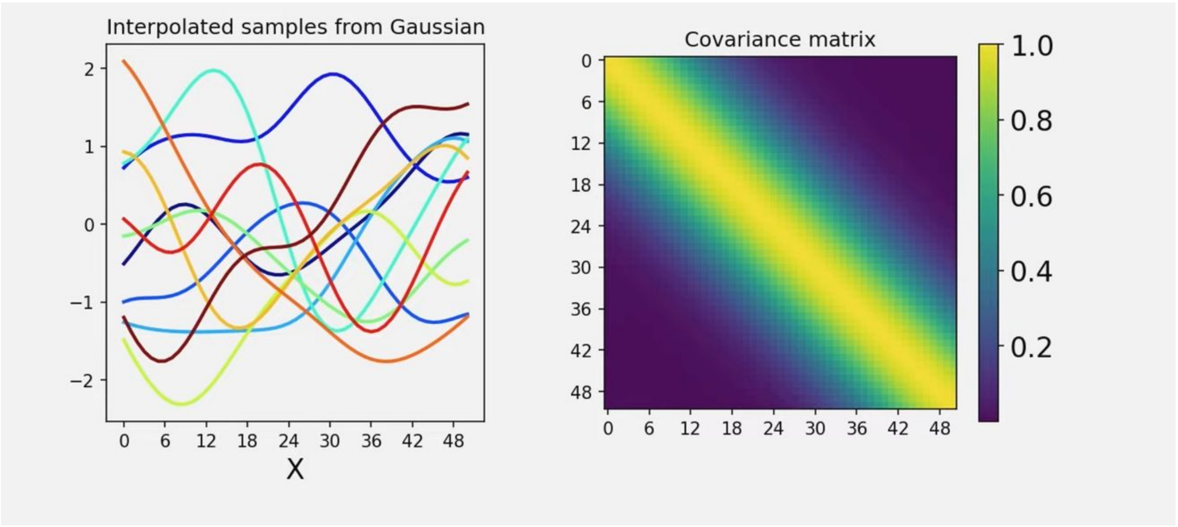
$$P(\mathbf{X}|\mathbf{w}) = \prod_{i=0}^{n} P(\mathbf{y}_i|\mathbf{x}_i;\mathbf{w}) = \prod_{i=0}^{n} \mathcal{N}(\mathbf{wx}, \sigma^2\mathbf{I})$$

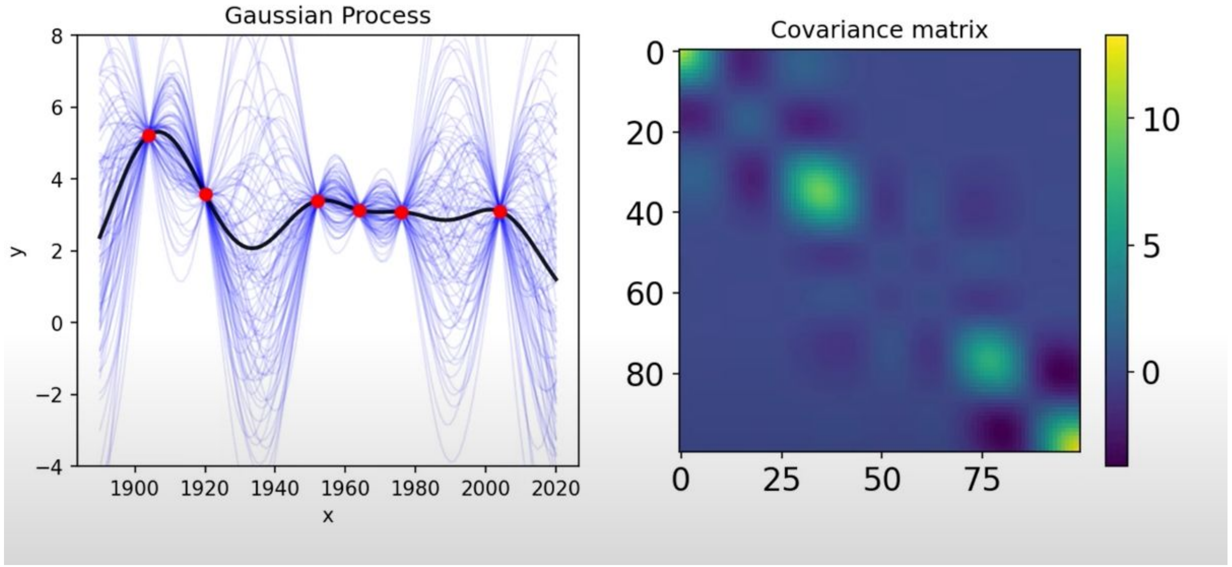




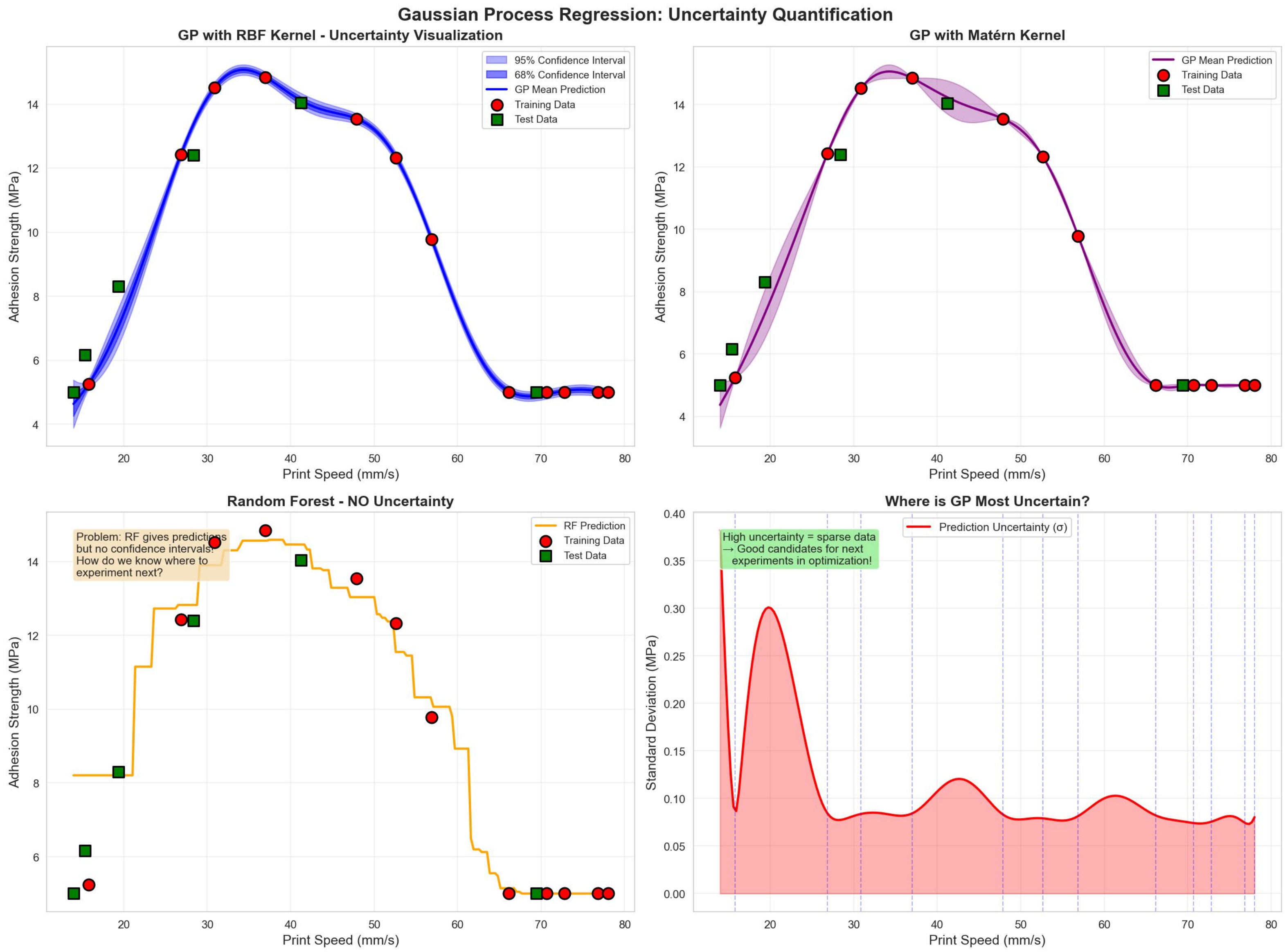
Ways to learn on the data with noise



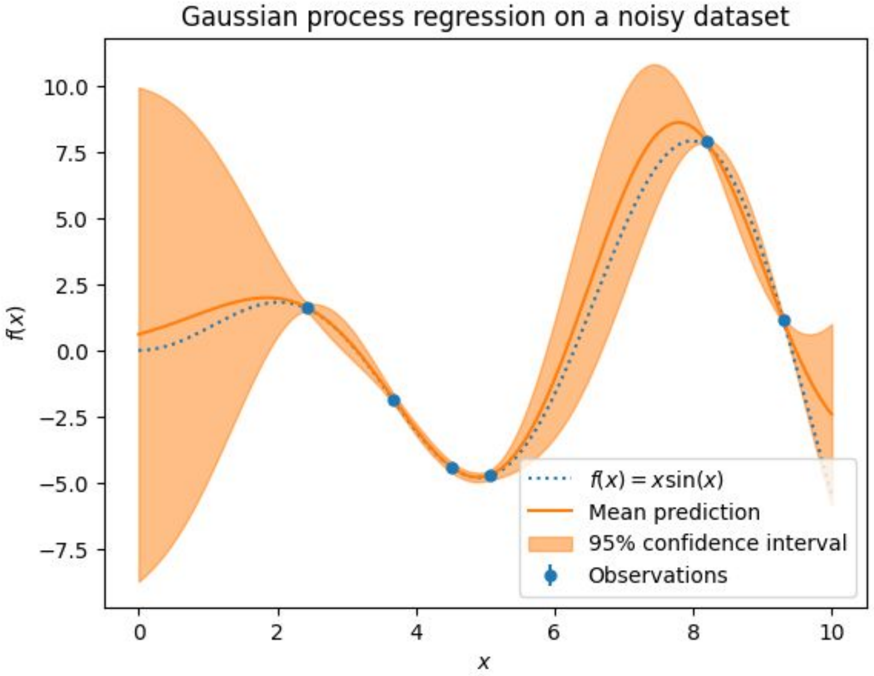


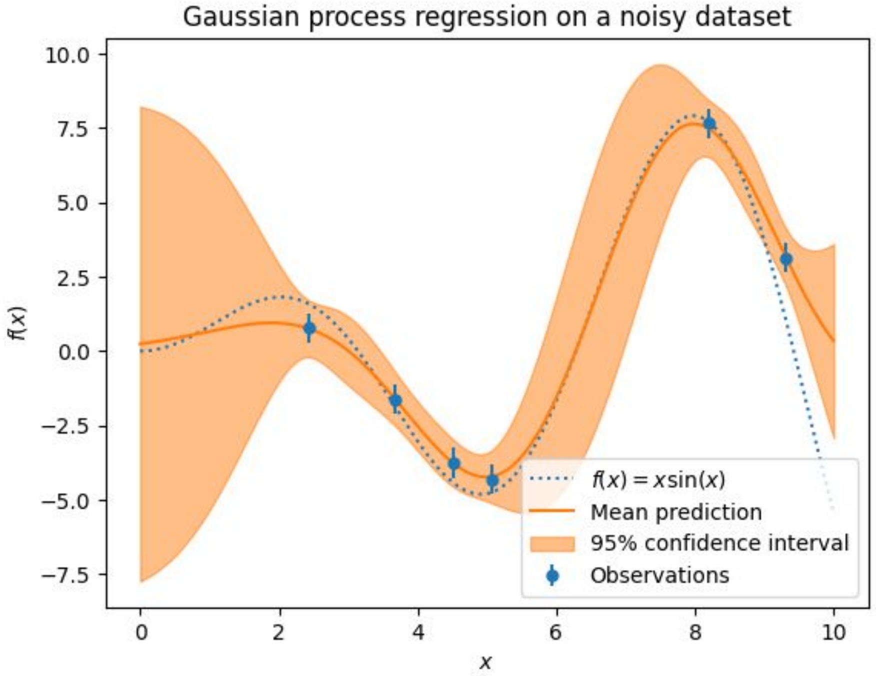


Demo2: Gaussian processes



Gaussian processes: noise





Gaussian processes: When to use?

Use when
- Small to medium datasets (10-1000 samples)
- Need uncertainty quantification
- Expensive experiments (want to minimize trials)
- Smooth or moderately complex relationships
- Optimization (Bayesian Optimization uses GP!)

Avoid when
- Very large datasets (>5000 samples) - too slow
- High-dimensional inputs (>20 features) - curse of dimensionality
- Discontinuous/chaotic relationships
- Real-time predictions needed - other models faster

But that does not solve the original problem!

Traditional approach: trial and error

- Test 5 params × 10 values each = 50 - 100000 experiments
- Misses parameter interactions
- Time-consuming and expensive

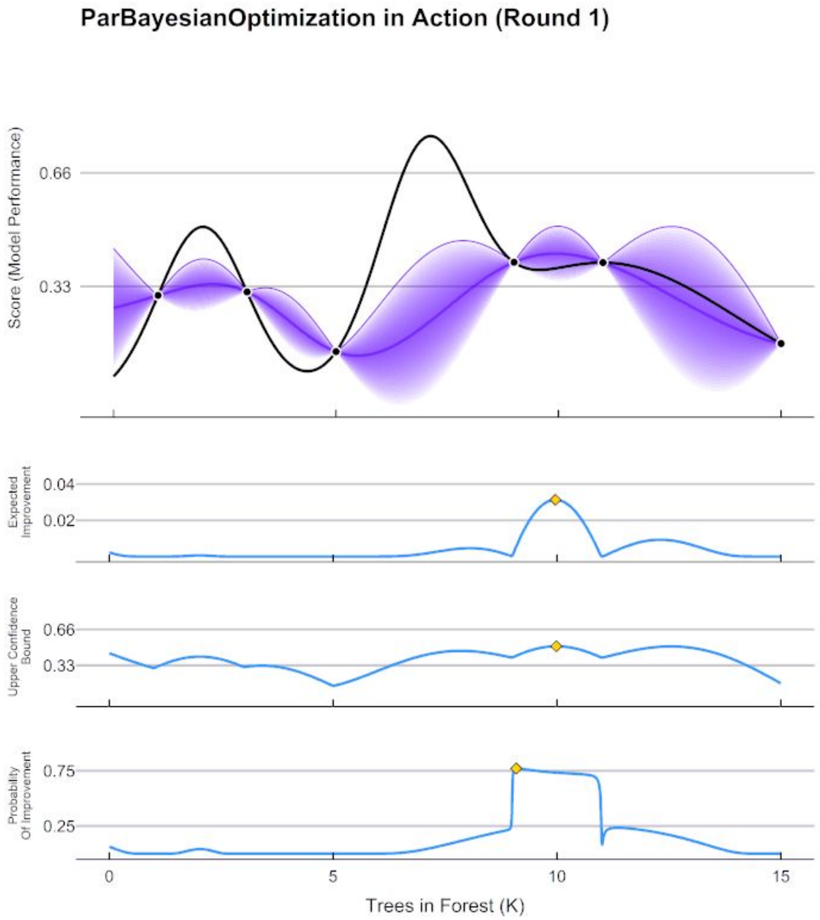
→

ML approach: Smart exploration

- Learn from fewer experiments
- Model parameter interactions
- Quantify uncertainty
- Guide next experiments intelligently

Bayesian optimization

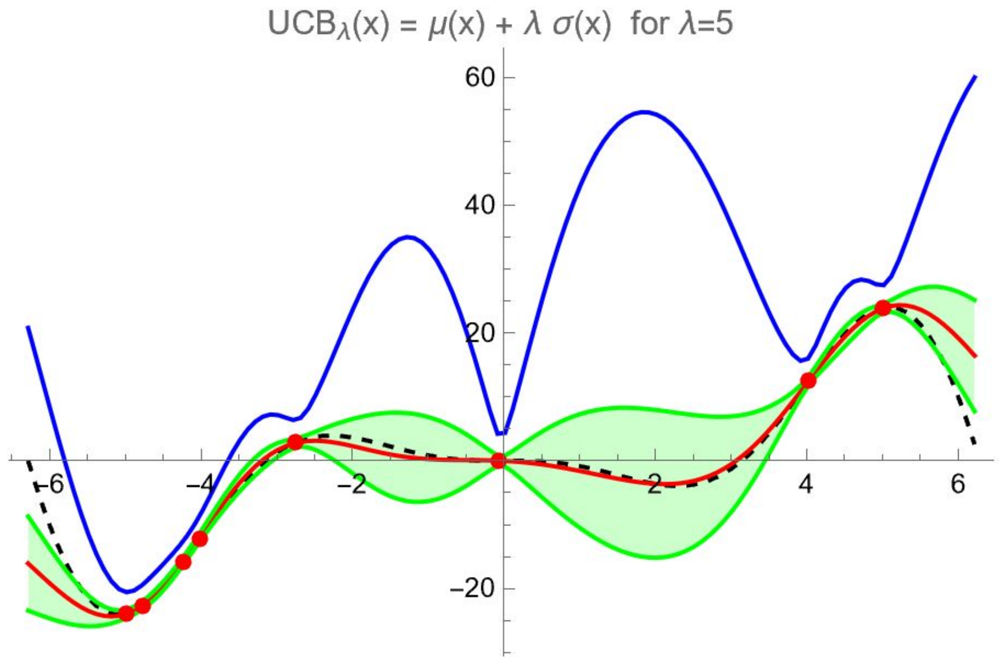
Step 1: Run a few initial experiments (5-10 random samples)
Step 2: Fit a Gaussian Process to the data
 - GP predicts outcome everywhere
 - GP gives uncertainty everywhere
Step 3: Use an acquisition function to decide the next experiment
 - Balance **exploitation** (try near current best)
 - vs **exploration** (try uncertain regions)
Step 4: Run the chosen experiment, add to the dataset
Step 5: Update GP, repeat Steps 3-4 until converged or budget exhausted
Result: Find the optimum in 20-100 experiments instead of thousands!

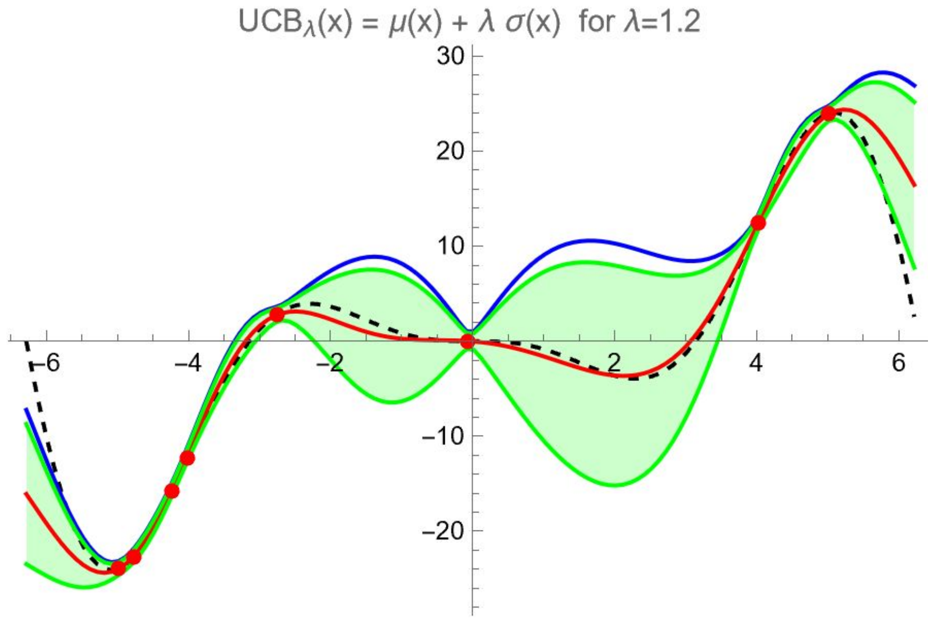


Acquisition functions - where to sample next?

Upper confidence bound (UCB) is a linear combination of the performance & uncertainty

$$a(x; \lambda) = \mu(x) + \lambda\sigma(x)$$





Acquisition functions - where to sample next?

Probability of improvement: PI(x) = P(f(x) > f(x_best))

- What's the probability this point is better than current best?
- Simple, but too greedy (over-exploits)

Expected improvement: EI(x) = E[max(f(x) - f(x_best), 0)]

- How much improvement do we expect?
- Good balance of exploration/exploitation
- Standard choice for most problems

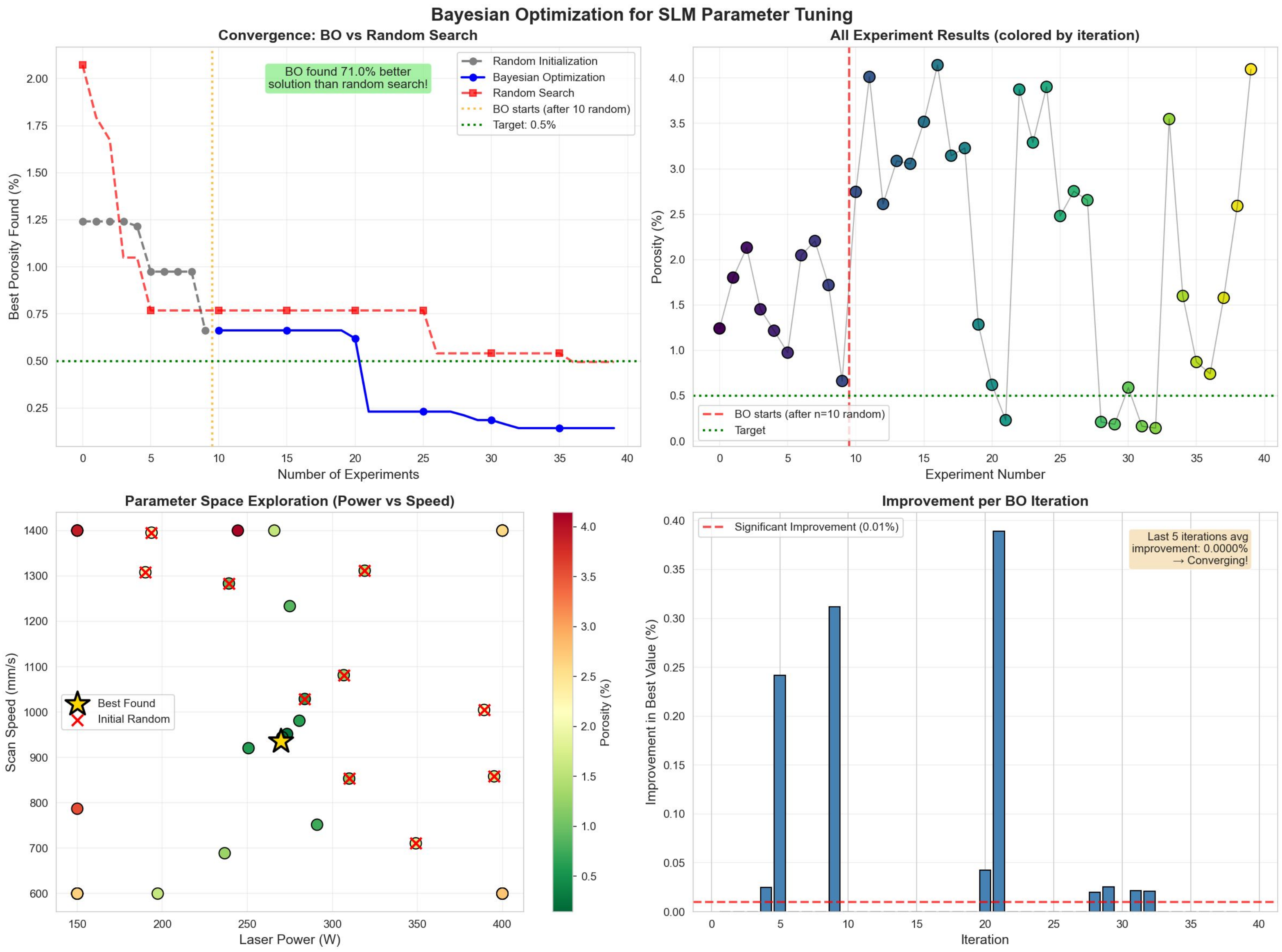
Thomson Sampling

- Sample a function from the GP posterior, optimize it
- Elegant theoretical properties
- Good for parallel optimization

Custom functions

- Usually based on UCB, when you want to define your own criteria

Demo 3: Bayesian optimization



Bayesian optimization: practical tips

Initial sampling strategy

- Use Latin Hypercube Sampling (LHS) for better space coverage
- Rule of thumb: n_initial = 2 × n_dimensions
- Example: 4 parameters → start with 8 experiments

Handling constraints

- Parameter bounds: always specify min/max for each parameter
- Linear constraints: use a constrained acquisition function
- Non-linear constraints: add penalty to acquisition function

Noisiy observations

- Real experiments have noise (measurement error, variability)
- Solution: Add noise parameter to GP (σ_n^2)
- Can repeat experiments at same point to reduce noise

Failures

- Some parameter combinations might fail (clogged nozzle, process crash)
- Strategy 1: Assign the worst possible value
- Strategy 2: Use robust Bayesian Optimization
- Strategy 3: Add failure prediction as a constraint

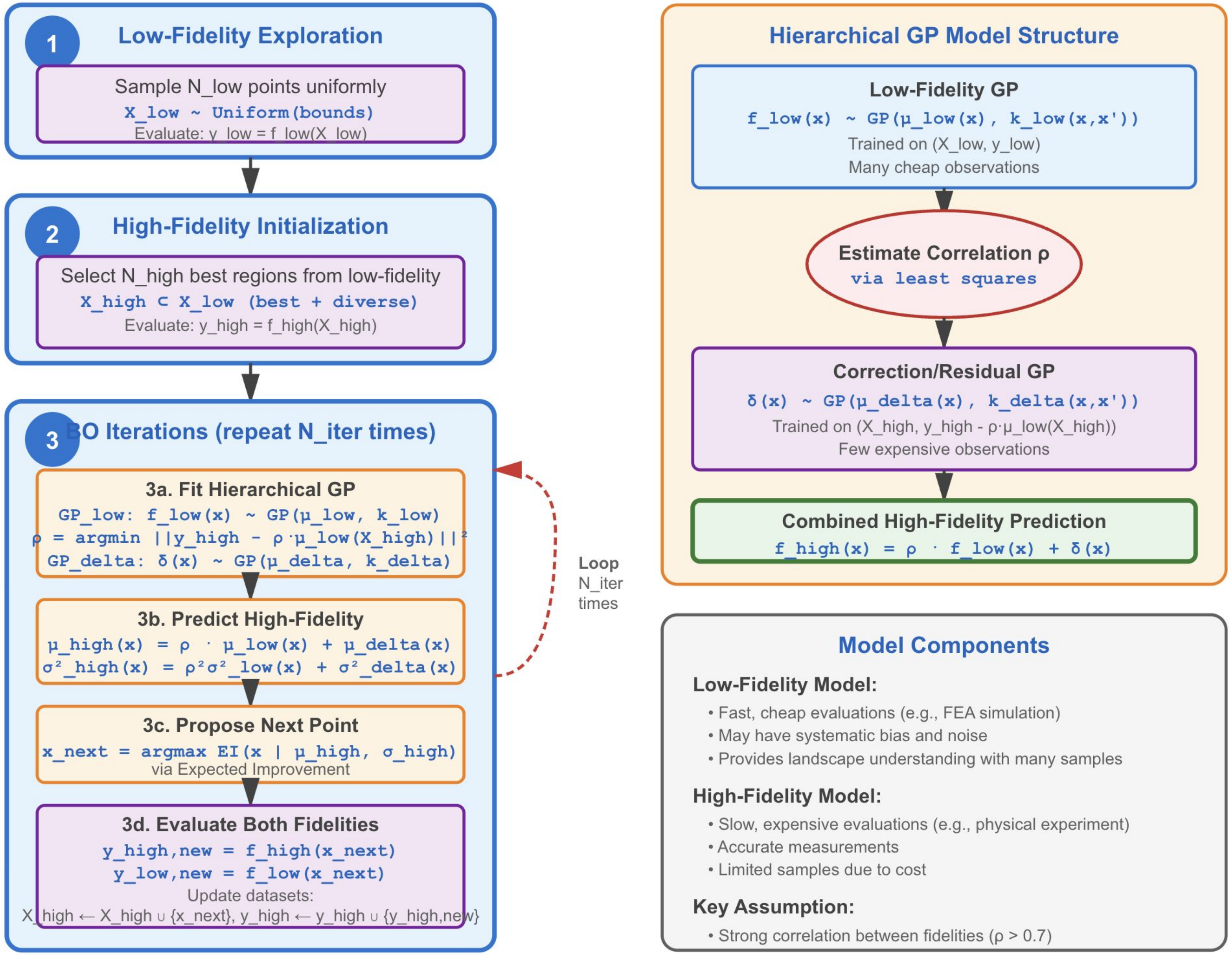
Multi-fidelity concept

High fidelity
- expensive, slow, accurate
- real printing & pricy materials

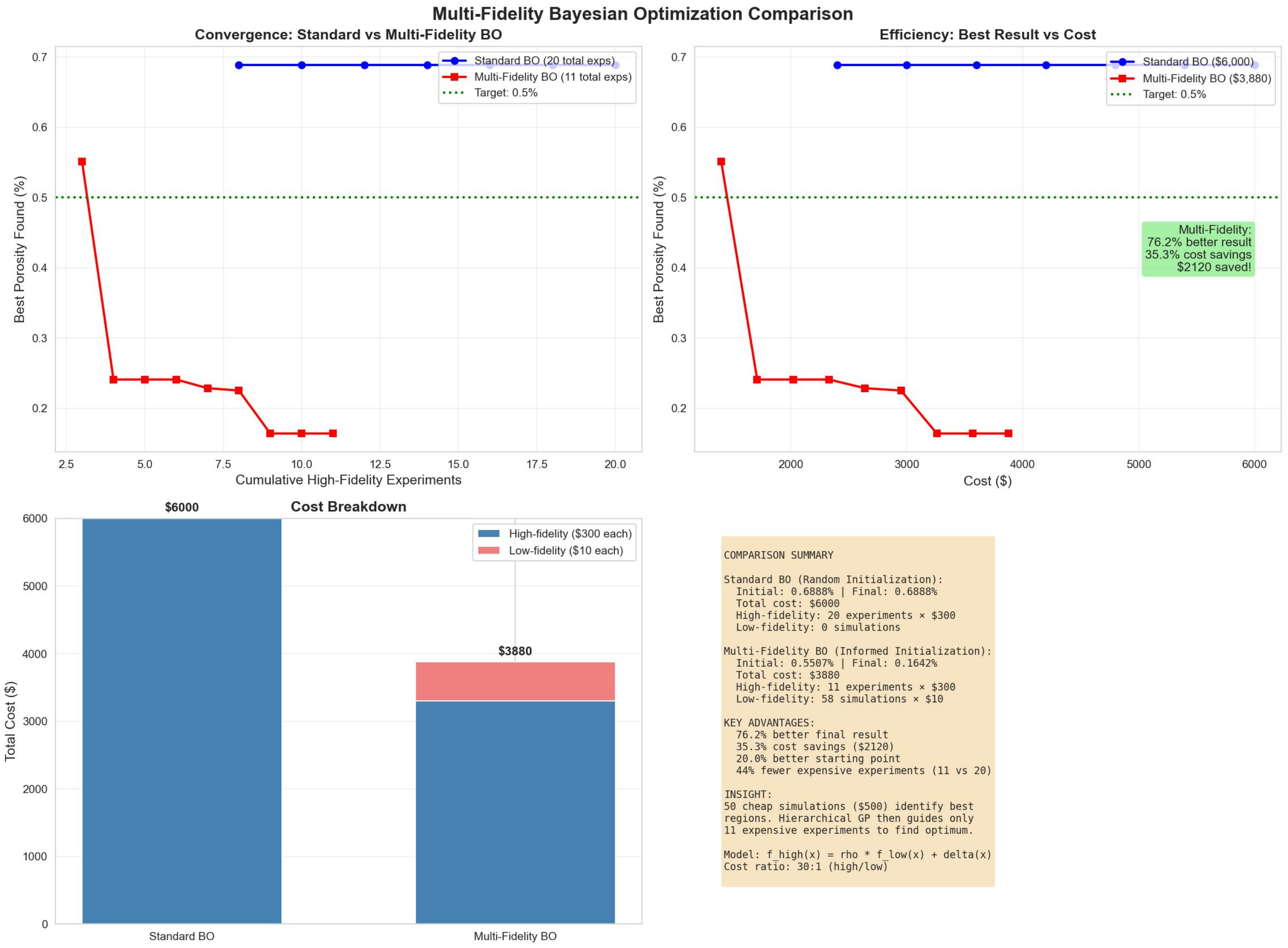
Low fidelity
- cheap, fast, approximate
- simulations, printing with cheap materials

Worflow
- Use MANY cheap simulations to learn general trends
- Use FEW expensive experiments to correct bias
- Combine both fidelitiess in a smart way
- **Result**: find optimum faster and cheaper

Multi-fidelity workflow



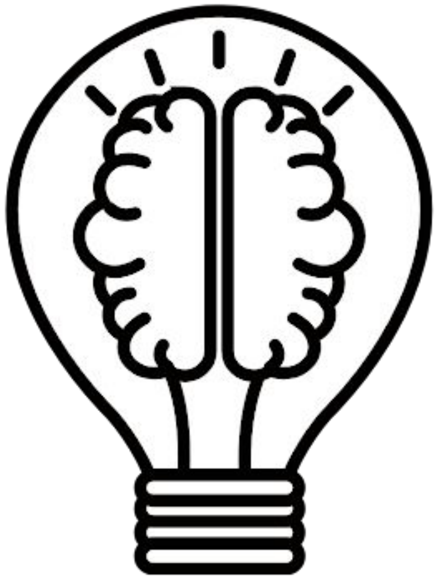
Demo 4: Multi-fidelity



Key takeaways

1. Regression is the foundation
- Many algorithms available (RF, GB, SVR, GP)
- Choose based on data size and uncertainty needs
- GP is critical when optimization is the goal

2. Uncertainty matters
- Knowing confidence intervals guides decisions
- Gaussian Processes provide principled uncertainty
- Essential for optimization and risk management



3. Bayesian optimization is powerful
- Find the optimum with 10-100× fewer experiments
- Balance exploration and exploitation
- Acquisition functions guide search intelligently

4. Multi-fidelity accelerates development even more
- Leverage cheap simulations + expensive experiments
- 2-5× speedup typical
- Requires good correlation between fidelities

Looking ahead to lecture 3: Advanced applications

We'll dive deeper into:

- Multi-objective Optimization: Optimize multiple conflicting goals simultaneously (quality vs speed vs cost)
- Deep Learning for AM: Computer vision for defect detection, physics-informed neural networks
- Real-time Process Control: Online monitoring and adaptive parameter adjustment
- Implementation Challenges: Deployment in production, integration with existing systems