

gradient variance $\propto \frac{1}{\text{batch size}}$; lower variance \Rightarrow larger learning rate

$$\begin{aligned} (\text{batch size})_{\text{new}} &\Rightarrow (\text{learning rate})_{\text{new}} \\ &= LR + LR \times \frac{1}{K} \\ \text{model capacity } \uparrow &\Rightarrow \text{regularisation } \uparrow \\ \text{small dataset} &\Rightarrow \text{aggressive regula-} \end{aligned}$$

$\left. \begin{array}{l} \\ \\ \end{array} \right\} \lambda \|\theta\|^2$

Optimisation: find optimal parameters to minimise predefined loss func.

Regularisation: add a penalty term to loss func during optimisation to prevent overfitting

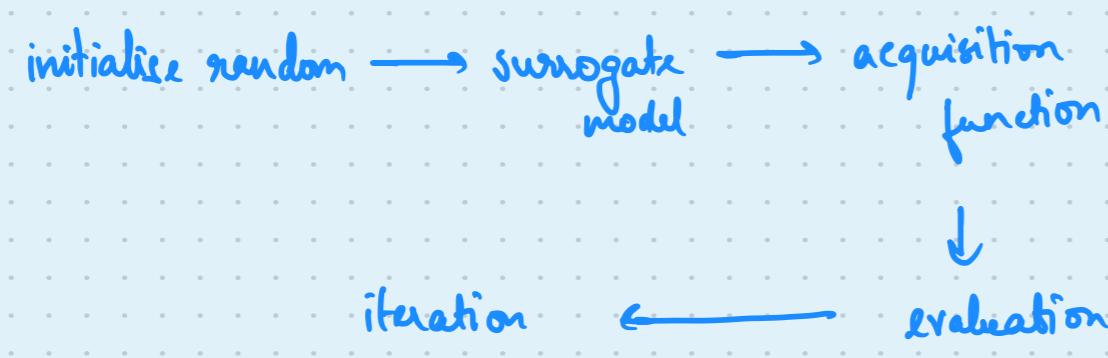
No Free Lunch Theorem

$$\sum_P F_A(P) = \sum_P F_B(P)$$

where P : set of all problems

Hyperparameter Optimisation

Bayesian / Sequential Model Based Optimisation



Say we use a Gaussian Process model for surrogate

We use a Gaussian Process (GP) prior.

- A GP gives us a predictive distribution at any point x :

$$M_t(x) \sim N(\mu_t(x), \sigma_t^2(x))$$

where

- $\mu_t(x)$ is the predicted mean (our best guess).
- $\sigma_t(x)$ is the uncertainty.

balance exploitation and exploration

common acquisition function: Expected Improvement

$$EI_{y^*}(x) = \int_{-\infty}^{\infty} \max(y^* - y, 0) P_m(y|x) dy$$

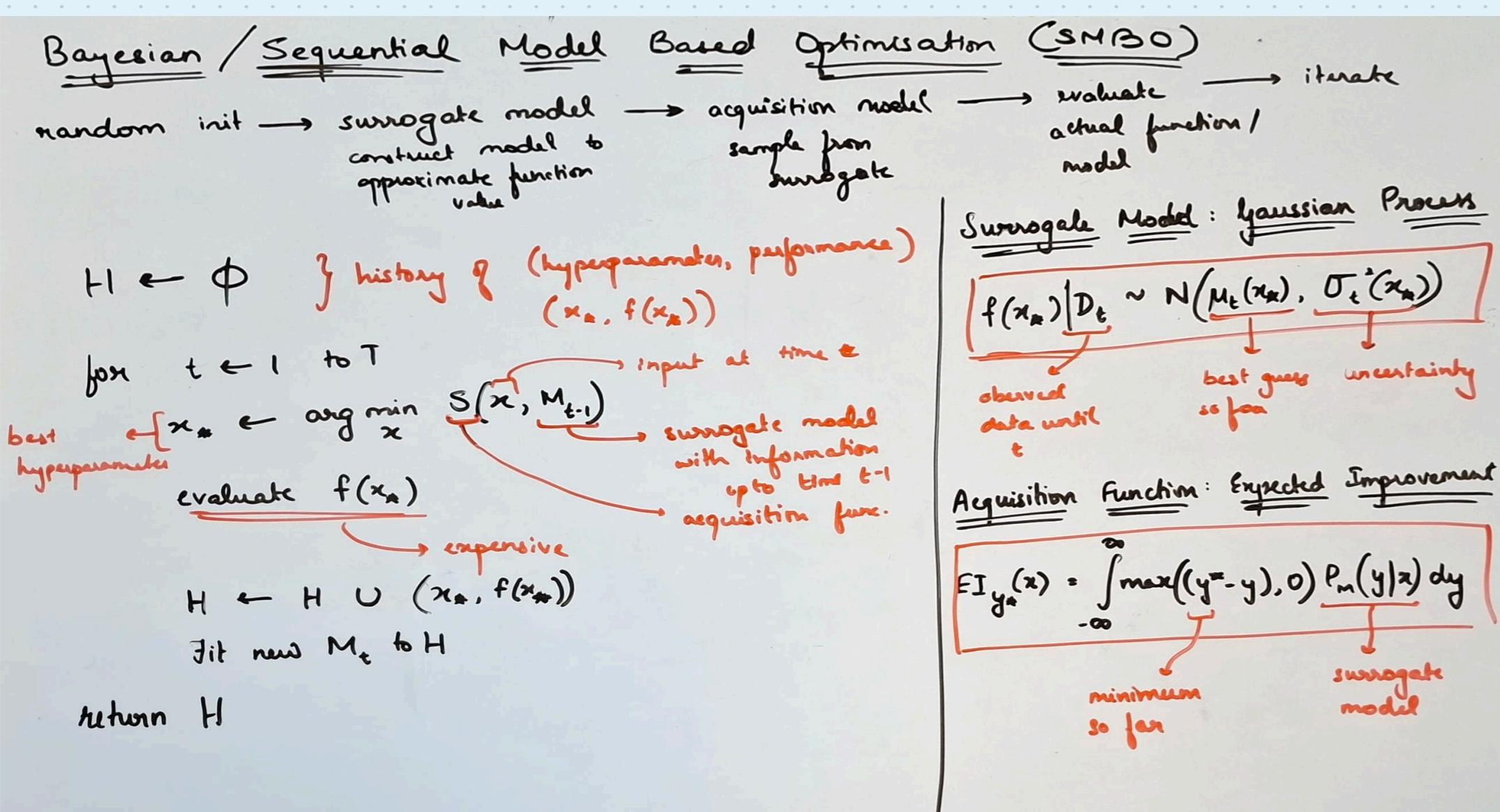
new obj. function score
hyperparameter
surrogate model
min so far

SMBO Algorithm

SMBO(f, M_0, T, S)

```

H ← ∅ // history of (hyperparameters, obj. func score)
for t ← 1 to T → max iterations
    next hyperparameter [x* ← argmin_x S(x, M_{t-1})] → acquisition function
    surrogate model: contains what we know so far, and our uncertainty
    find some x for S that minimises S
    Evaluate f(x*) // expensive
    H ← H ∪ (x*, f(x*))
    Fit new M_t to H
return H
    
```



Hyperparameters in common ML models

Decision tree: max depth, leaves per split

K-NN, K-means: K

Random Forest: no. of trees

Naive Bayes: smoothing parameters → probabilities multiplied,

small constant to prevent zeroes when data not present in training

logistic Regression: solver

SVM: kernel → linear
poly → Radial Basis Function (RBF)

Hidden Markov: epochs, length of observations, hidden states

Gaussian Mixture: no. of Gaussians

Neural networks: learning rate, no. of epochs, batch size, step

decay: time-based, step-based, cosine
adaptive: ADAM, Adagrad, RMSprop

step = forward + backward + update

Hyperparameters Interaction Web

Nothing exists in isolation

learning rate ← batch size ← architecture
↓ ↓ ↓
Regularisation ← dropout rate ← weight decay