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

$(\text{batch size})_{new} = BS + BS \times K \Rightarrow (\text{learning rate})_{new} = LR + LR \times \sqrt{K}$

model capacity ↑ $\Rightarrow$ regularisation ↑

small dataset $\Rightarrow$ aggressive regula.

$\times \|\theta\|^2$

**Optimisation**: find optimal parameters to minimise predefined loss func.

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

# Hyperparameter Optimisation

## No Free Lunch Theorem

$\sum_P E_A(P) = \sum_P E_B(P)$

where P: set of all problems

## Bayesian / Sequential Model Based Optimisation

initialise random → surrogate model → acquisition function → evaluation → iteration

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 $x$:

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

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

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

$y^*$: min so far; $y$: new obj. function score; $x$: hyperparameter; $P_m(y|x)$: surrogate model

## SMBO Algorithm

SMBO($f, M_0, T, S$)

$H \leftarrow \phi$ // history of (hyperparameter, obj. func score)

for $t \leftarrow 1$ to $T$ ] → max iterations

next hyperparameter to evaluate ← [ $x^* \leftarrow \text{argmin}_x S(x, M_{t-1})$

$S$: acquisition function; $M_{t-1}$: surrogate model: contains what we know so far, and our uncertainty; $x$: input

argmin: find some $x$ for $S$ that minimises $S$

Evaluate $f(x^*)$ // expensive

$H \leftarrow H \cup (x^*, f(x^*))$

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 parameter → probabilities multiplied, small constant to prevent zeroes when data not present in training

$O(n^3)$ 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 → no of batches processed in one epoch

step = forward + backward + update

## Hyperparameter Interaction Web

Nothing exists in isolation

learning rate ↔ batch size ↔ architecture depth

Regularisation ↔ dropout rate ↔ weight decay