Các phương pháp tính toán: Từ toán học cổ điển đến trí tuệ nhân tạo

Ngo Tung Son (Ph.D)

Dean, Faculty of Information Assurance, FPT University

Mở đầu

Các phương pháp tính toán tạo thành nền tảng để giải quyết bài toán trong toán học ứng dụng, khoa học máy tính, kỹ thuật và trí tuệ nhân tạo (Artificial Intelligence). Mỗi phương pháp có một giả định riêng về dữ liệu, không gian nghiệm, mức độ chính xác cần đạt và chi phí tính toán có thể chấp nhận.

Tài liệu này trình bày các nhóm phương pháp theo một mạch thống nhất. Phương pháp tất định (Deterministic) cung cấp nghiệm chính xác khi bài toán có cấu trúc rõ ràng. Phương pháp ngẫu nhiên (Stochastic) cho phép xử lý bài toán lớn hoặc khó mô hình hóa đầy đủ. Phương pháp xấp xỉ (Approximation) chấp nhận sai số có kiểm soát để đổi lấy khả năng tính toán thực tế. Nội suy và ngoại suy (Interpolation and Extrapolation) giúp suy luận từ dữ liệu hữu hạn. Tối ưu hóa (Optimization) đóng vai trò cơ chế tìm nghiệm. Học máy (Machine Learning) và AI kết hợp nhiều cơ chế trên trong một hệ thống học từ dữ liệu.

Mục tiêu của tài liệu không phải là liệt kê thuật toán rời rạc. Trọng tâm là làm rõ logic toán học phía sau từng nhóm phương pháp, điều kiện áp dụng, giới hạn kỹ thuật và lý do vì sao các phương pháp hiện đại, đặc biệt là AI, có thể hoạt động hiệu quả trên dữ liệu thực tế.

I. Phương pháp tất định (Deterministic)

Phần này trình bày nhóm thuật toán có hành vi hoàn toàn xác định. Đây là điểm xuất phát quan trọng vì nó cho thấy khi nào bài toán có thể được giải chính xác, khi nào chi phí tính toán bắt đầu vượt quá khả năng xử lý thực tế và vì sao các phương pháp khác phải xuất hiện.

Một thuật toán là tất định (Deterministic) khi cùng một đầu vào luôn tạo ra cùng một đầu ra theo cùng một chuỗi bước xử lý. Thuật toán không sử dụng yếu tố ngẫu nhiên trong quá trình tính toán.

Định nghĩa hình thức Thuật toán $\mathcal{A}$ là tất định nếu tồn tại hàm $f: \mathcal{X} \to \mathcal{Y}$ sao cho $\forall x \in \mathcal{X}: \mathcal{A}(x) = f(x)$ với kết quả hoàn toàn xác định.

Ví dụ 1: Giải hệ phương trình tuyến tính (Gaussian Elimination)

Bài toán: tìm $\mathbf{x} \in \mathbb{R}^n$ thoả $A\mathbf{x} = \mathbf{b}$ với $A \in \mathbb{R}^{n\times n}$, $\det(A) \neq 0$.

Phép khử Gauss: LU Decomposition

$$A = LU \quad \Rightarrow \quad L\mathbf{y} = \mathbf{b} \overset{\text{forward}}{\rightarrow} \mathbf{y} \overset{\text{back}}{\rightarrow} U\mathbf{x} = \mathbf{y}$$

Trong đó L là ma trận tam giác dưới, U là tam giác trên. Nghiệm duy nhất và chính xác:

$$\mathbf{x} = A^{-1}\mathbf{b} = U^{-1}L^{-1}\mathbf{b}$$

Phân tích độ phức tạp: Phép khử Gauss tốn $\mathcal{O}(n^3)$ phép toán. Với $n = 1000$: khoảng 10^9 phép tính: tất định và chính xác tuyệt đối.

Ví dụ 2: Thuật toán Dijkstra

Tìm đường đi ngắn nhất từ nguồn s đến tất cả đỉnh trong đồ thị $G = (V, E, w)$ với $w \geq 0$.

Invariant của Dijkstra

$$d[v] = \min_{p: s \rightsquigarrow v} \sum_{(u,w)\in p} w\,(u, w)$$

Khi đỉnh v được extract khỏi priority queue, $d[v]$ là chính xác tuyệt đối. Chứng minh bằng quy nạp trên tập S (đã xét).

$$\forall v \in S: d[v] = \delta(s, v) \quad \text{(khoảng cách thực)}$$

Thách thức của Deterministic là Curse of dimensionality: Không gian bài toán tăng theo hàm mũ: $|\mathcal{X}| = O(k^n)$. Giải chính xác mọi bài toán tổ hợp n-chiều là không khả thi. Ví dụ: Traveling Salesman Problem (TSP) với n thành phố: không gian tìm kiếm $(n-1)!/2$, thuộc lớp NP hard.

- Bài toán phù hợp: Hệ phương trình tuyến tính, đường đi ngắn nhất, sắp xếp, tìm kiếm nhị phân, FFT, đồ thị DAG.
- Không phù hợp: Bài toán NP hard, không gian trạng thái liên tục $\mathbb{R}^n$ chiều cao, dữ liệu nhiễu, bài toán học từ dữ liệu.

II. Phương pháp ngẫu nhiên (Stochastic)

Phần này xem xét các phương pháp đưa xác suất vào quá trình tính toán. Cách tiếp cận này phù hợp khi không thể duyệt hết không gian trạng thái, không thể tính tích phân trực tiếp hoặc cần lấy mẫu từ một phân phối phức tạp. Giá trị của stochastic nằm ở khả năng mở rộng, còn giới hạn chính nằm ở phương sai, thời gian hội tụ và độ tin cậy xác suất.

Phương pháp ngẫu nhiên (Stochastic) sử dụng yếu tố xác suất trong quá trình tính toán. Kết quả không còn được xác định tuyệt đối bởi đầu vào, nhưng phương pháp này phù hợp với các bài toán có không gian lớn, nhiều chiều hoặc khó mô hình hóa bằng thuật toán tất định.

Nền tảng toán học: Luật số lớn và CLT

Luật số lớn Mạnh (SLLN) Cho $X_1, X_2, \ldots$ iid với $\mathbb{E}[|X_1|] < \infty$. Khi đó:

$$\frac{1}{N}\sum_{i=1}^{N} X_i \xrightarrow{\text{a.s.}} \mathbb{E}[X_1] \quad \text{khi } N \to \infty$$

Central Limit Theorem (CLT)

$$\sqrt{N}(\bar{X}_N - \mu) \xrightarrow{d} \mathcal{N}(0, \sigma^2)$$

Sai số của ước lượng Monte Carlo tỉ lệ $\sim \sigma/\sqrt{N}$: không phụ thuộc chiều d! Đây là điểm quan trọng.

Ví dụ 1: Monte Carlo Integration

Tính $I = \int_{\Omega} f(\mathbf{x})\, d\mathbf{x}$ với $\Omega \subset \mathbb{R}^d$ phức tạp.

Monte Carlo Estimator

$$I = |\Omega| \cdot \mathbb{E}_{\mathbf{x}\sim\text{Uniform}(\Omega)}[f(\mathbf{x})] \approx \frac{|\Omega|}{N}\sum_{i=1}^{N} f(\mathbf{x}_i), \quad \mathbf{x}_i \sim \text{Uniform}(\Omega)$$

Sai số chuẩn:

$$\text{SE} = \frac{|\Omega| \cdot \sigma_f}{\sqrt{N}}, \quad \sigma_f^2 = \text{Var}[f(\mathbf{x})]$$

Phương pháp bậc thang (quadrature) tốn $O(N^d)$ điểm cho sai số $O(N^{-2/d})$. Monte Carlo: sai số $O(N^{-1/2})$ bất kể d.

Ví dụ 2: MCMC (Markov Chain Monte Carlo)

Lấy mẫu từ phân phối $\pi(\mathbf{x}) \propto p(\mathbf{x})$ khi p không tính chuẩn hoá được.

Metropolis Hastings

$$\alpha(\mathbf{x}, \mathbf{x}') = \min\left(1, \frac{\pi(\mathbf{x}')\, q(\mathbf{x}|\mathbf{x}')}{\pi(\mathbf{x})\, q(\mathbf{x}'|\mathbf{x})}\right)$$

Chuỗi Markov $\{X_t\}$ hội tụ đến π nếu thoả detailed balance:

$$\pi(\mathbf{x})\, T(\mathbf{x}'|\mathbf{x}) = \pi(\mathbf{x}')\, T(\mathbf{x}|\mathbf{x}')$$

Ergodic Theorem: $\frac{1}{T}\sum_{t=1}^{T} g\,(X_t) \overset{a.s.}{\to} \mathbb{E}_\pi[g(X)]$

Ví dụ 3: Simulated Annealing (Tối ưu ngẫu nhiên)

Xác suất chấp nhận trạng thái xấu hơn

$$P(\text{accept}) = \exp\left(-\frac{\Delta E}{T(k)}\right), \quad T(k) = T_0 \cdot \alpha^k,\ \alpha \in (0.9,\ 1)$$

Khi $T \to 0$: xác suất chấp nhận trạng thái xấu $\to 0$: thuật toán hội tụ dần về nghiệm tốt. Chứng minh hội tụ đến tối ưu toàn cục với lịch làm nguội loga: $T(k) = c/\ln(1+k)$.

Thách thức Stochastic Variance: Nếu Var$[f]$ lớn, cần N rất lớn để đạt độ chính xác. Mixing time: MCMC có thể hội tụ chậm nếu phân phối đa đỉnh (multimodal). Không có thuật toán tốt nhất cho mọi bài toán: Đổi determinism lấy scalability: đúng với xác suất cao, không chắc chắn tuyệt đối.

III. Phương pháp xấp xỉ (Approximation)

Phần này tập trung vào cách thay thế nghiệm chính xác bằng nghiệm đủ tốt có cơ sở toán học. Đây là hướng tiếp cận quan trọng đối với các bài toán tối ưu tổ hợp, bài toán NP hard và các phép tính số không thể biểu diễn chính xác trong thực tế. Điều cần quan tâm không chỉ là sai số, mà là sai số đó có được kiểm soát hay không.

Phương pháp xấp xỉ (Approximation) không đặt mục tiêu tìm nghiệm chính xác tuyệt đối trong mọi trường hợp. Mục tiêu chính là tìm nghiệm đủ tốt trong thời gian chấp nhận được, đồng thời có thể nêu rõ sai số hoặc tỉ số bảo đảm.

Approximation Algorithm: Lý thuyết

Approximation Ratio Thuật toán $\mathcal{A}$ là α-approximation cho bài toán tối thiểu hoá nếu:

$$\forall \text{ instance } I: \quad \mathcal{A}(I) \leq \alpha \cdot \text{OPT}(I), \quad \alpha \geq 1$$

Ví dụ: Vertex Cover (2 approximation)

Chứng minh tỉ số 2

Gọi M là matching cực đại tìm được. Mọi cạnh trong M cần ít nhất 1 đỉnh cover. Thuật toán chọn cả 2 đầu mút của mỗi cạnh trong M:

$$|C| = 2|M| \leq 2 \cdot \text{OPT} \quad \Rightarrow \quad \alpha = 2$$

Vì OPT cần ít nhất $|M|$ đỉnh để cover M cạnh rời nhau.

Numerical Approximation: Taylor và Padé

Taylor Series + Error Bound

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k + R_n(x)$$

Lagrange remainder:

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-a)^{n+1}, \quad \xi \in (a,x)$$

Ví dụ: e^x xấp xỉ bậc 4 tại $x = 1$:

$$e^1 \approx 1 + 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} = \frac{65}{24} \approx 2.7083, \quad |R_4| \leq \frac{e}{120} \approx 0.0227$$

Approximation qua Fourier: Tín hiệu và Nén

Best k term Approximation

$$f(x) \approx \hat{f}_k(x) = \sum_{|j| \leq k} c_j \, e^{2\pi i j x/T}, \quad c_j = \frac{1}{T}\int_0^T f\,(x) e^{-2\pi i j x/T} dx$$

Theo Parsevals theorem:

$$\| f - \hat{f}_k \|_{L^2}^2 = \sum_{|j|>k} |c_j|^2$$

Với tín hiệu âm thanh/hình ảnh, năng lượng tập trung ở tần số thấp dẫn đến nén JPEG/MP3 loại bỏ hệ số nhỏ mà lỗi tri giác thấp.

Thách thức Approximation Inapproximability: Một số bài toán NP hard có tỉ số xấp xỉ tốt nhất là $O(\log n)$ (Set Cover). Một số không thể xấp xỉ tốt hơn tỉ lệ nhất định trừ phi $P = NP$ (định lý PCP). Bias variance: Xấp xỉ bậc thấp dẫn đến bias lớn; bậc cao dẫn đến không ổn định (Runges phenomenon).

IV. Nội suy và ngoại suy (Interpolation and Extrapolation)

Phần này trình bày cách xây dựng hàm từ một tập quan sát hữu hạn. Interpolation phù hợp khi cần tái tạo giá trị trong miền dữ liệu đã biết. Extrapolation khó hơn vì phải dự đoán ngoài vùng quan sát. Do đó, phần này cần được đọc cùng với các khái niệm về sai số, ổn định số và mức độ không chắc chắn.

Cho tập điểm dữ liệu $\{(x_i, y_i)\}_{i=0}^n$, tìm hàm $p(x)$ sao cho $p(x_i) = y_i$ với mọi i (interpolation) hoặc ước lượng f ngoài phạm vi dữ liệu (extrapolation).

1. Nội suy Lagrange

Đa thức Lagrange bậc n

$$p_n(x) = \sum_{i=0}^{n} y_i \; L_i(x), \quad L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

Sai số nội suy:

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i), \quad \xi \in [\min x_i, \max x_i]$$

Hiện tượng Runge: với $f(x) = 1/(1 + 25x^2)$, điểm nút đều dẫn đến lỗi tăng mạnh ở biên khi $n \to \infty$.

2. Spline Cubic: Giải pháp thực tế

Cubic Spline: Điều kiện C^2 liên tục

Trên mỗi đoạn $[x_i, x_{i+1}]$, dùng đa thức bậc 3:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Điều kiện:

$$S_i(x_{i+1}) = S_{i+1}(x_{i+1}), \quad S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}), \quad S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$$

Dẫn đến hệ tridiagonal $4n \times 4n$: giải được bằng Thomas algorithm trong $\mathcal{O}(n)$. Đây là lý do spline được dùng trong thiết kế CAD, font chữ, đồ hoạ vector.

3. Kriging / Gaussian Process Interpolation

Phiên bản stochastic của nội suy: cung cấp cả giá trị dự đoán lẫn độ không chắc chắn.

GP Posterior

$$f(\mathbf{x}) \sim \mathcal{GP}\big(m(\mathbf{x}),\, k(\mathbf{x}, \mathbf{x}')\big)$$

Posterior sau khi quan sát $\mathbf{y} = \mathbf{f} + \boldsymbol{\varepsilon}$:

$$\mu_*(\mathbf{x}) = m(\mathbf{x}) + K_{*X}(K_{XX} + \sigma^2 I)^{-1}\big(\mathbf{y} - m(\mathbf{X})\big)$$

$$\sigma_*^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - K_{*X}(K_{XX} + \sigma^2 I)^{-1} K_{X*}$$

Tại điểm đã quan sát: $\sigma_* = 0$ (nội suy chính xác). Ngoài vùng dữ liệu: σ_* tăng dẫn đến biểu diễn được mức độ không chắc chắn.

Thách thức Interpolation Runges Phenomenon: Đa thức bậc cao với nút đều không ổn định ở biên. Overfitting: Đi qua mọi điểm ≠ mô hình tốt khi dữ liệu có nhiễu. GP scalability: Tốn $O(n^3)$ do phải nghịch đảo K_{XX}, hạn chế với $n > 10^4$.

V. Tối ưu hóa (Optimization)

Phần này trình bày optimization như cơ chế trung tâm của tính toán hiện đại. Nhiều bài toán không chỉ yêu cầu tính một giá trị, mà yêu cầu tìm cấu hình tốt nhất theo một hàm mục tiêu. Trong học máy, optimization quyết định cách mô hình điều chỉnh tham số để giảm sai số trên dữ liệu huấn luyện.

Nền tảng của mọi phương pháp: đặc biệt ML. Bài toán tổng quát: $\min_{\mathbf{x} \in \mathcal{C}} f(\mathbf{x})$.

Gradient Descent và biến thể

Cập nhật Gradient Descent

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \nabla f(\mathbf{x}_t)$$

Hội tụ với f L-smooth và μ-strongly convex:

$$f(\mathbf{x}_T) - f^* \leq \left(1 - \frac{\mu}{L}\right)^T (f(\mathbf{x}_0) - f^*)$$

Tốc độ hội tụ tuyến tính với $\eta = 1/L$. Condition number $\kappa = L/\mu$ quyết định tốc độ.

SGD (Stochastic Gradient Descent): thành phần quan trọng của deep learning

$$\nabla f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla \ell(\mathbf{x}; z_i) \approx \frac{1}{b}\sum_{j \in \mathcal{B}} \nabla \ell\left(\mathbf{x}; z_j\right)$$

Mỗi bước tốn $O(b)$ thay vì $O(n)$. Với $b \ll n$: tốc độ mỗi bước cao hơn n/b lần.

Hội tụ SGD (non convex, Lipschitz gradient):

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[\| \nabla f(\mathbf{x}_t) \|^2\right] \leq \frac{2(f(\mathbf{x}_0) - f^*)}{\eta T} + \eta L \sigma^2$$

Chọn $\eta = O\left(1/\sqrt{T}\right)$: $\min_t \mathbb{E}[\| \nabla f \|^2] = O\left(1/\sqrt{T}\right)$.

Adam Optimizer: Adaptive Moments

Adam: Bias corrected adaptive learning rate

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad \text{(1st moment)}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad \text{(2nd moment)}$$

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \widehat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad \text{(bias correction)}$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \frac{\eta}{\sqrt{\widehat{v}_t} + \epsilon} \widehat{m}_t$$

Mỗi tham số có learning rate riêng, thích nghi theo lịch sử gradient. $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ là giá trị mặc định.

VI. Học máy (Machine Learning) dưới góc nhìn toán học

Phần này đặt Machine Learning trong quan hệ trực tiếp với các phương pháp đã trình bày trước đó. Machine Learning không tách rời toán học cổ điển. Nó kết hợp approximation để biểu diễn hàm, optimization để tìm tham số, stochastic để học trên dữ liệu lớn và interpolation để tổng quát hóa từ mẫu quan sát.

Học máy (Machine Learning) là sự kết hợp có hệ thống của nhiều phương pháp tính toán. Neural network đóng vai trò như mô hình xấp xỉ, gradient descent cung cấp cơ chế tối ưu, SGD đưa yếu tố ngẫu nhiên vào huấn luyện, còn kernel methods và Gaussian Process liên hệ trực tiếp với nội suy.

1. Học Có Giám Sát: Statistical Learning Theory

Risk và Empirical Risk

$$R(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f(x), y)] \quad \text{(True Risk - không biết)}$$

$$\hat{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(f(x_i), y_i\right) \quad \text{(Empirical Risk - tính được)}$$

Mục tiêu: $\min_{f\in\mathcal{H}} \hat{R}_n(f)$ sao cho $R(f) \approx \hat{R}_n(f)$.

PAC Learning Bound: VC Theory

Với xác suất $\geq 1-\delta$, với mọi $f \in \mathcal{H}$:

$$R(f) \leq \hat{R}_n(f) + \sqrt{\frac{8 \cdot \mathrm{VC}(\mathcal{H}) \cdot \ln\frac{en}{\mathrm{VC}(\mathcal{H})} + 8\ln\frac{4}{\delta}}{n}}$$

Ý nghĩa: Gap giữa train error và test error phụ thuộc vào $\mathrm{VC}(\mathcal{H})$ (độ phức tạp mô hình) và n (số mẫu). Training error thấp chưa đủ: cần $n \gg \mathrm{VC}(\mathcal{H})$.

2. Neural Network: Universal Approximation

Universal Approximation Theorem (Cybenko 1989, Hornik 1991) Cho hàm kích hoạt σ liên tục, bị chặn, không đa thức. Với mọi $f \in C\left([0,1]^d\right)$ và $\varepsilon > 0$, tồn tại mạng 1 lớp ẩn:

$$\left\| f - \sum_{j=1}^{N} \alpha_j\, \sigma\left(\mathbf{w}_j^T \mathbf{x} + b_j\right) \right\|_\infty < \varepsilon$$

Nhưng tồn tại không nói N bao nhiêu và làm sao tìm $\alpha_j, \mathbf{w}_j, b_j$. Deep network giải quyết điều này:

Deep Network: Representation Power

$$h^{(l)} = \sigma\left(W^{(l)} h^{(l-1)} + b^{(l)}\right), \quad l = 1, \ldots, L$$

Lợi thế của depth: Mạng L lớp, mỗi lớp n units biểu diễn được hàm cần $O(n^L)$ units trong mạng 1 lớp. Các hàm như $f(x) = \sin(2^n x)$ cần $\Theta(2^n)$ neurons nếu 1 lớp, chỉ cần $O(n)$ neurons nếu n lớp (bằng cách tổ hợp sin hai lần).

3. Backpropagation: Chuỗi đạo hàm (Chain Rule)

Backprop là Chain Rule được cài đặt hiệu quả

$$\frac{\partial \mathcal{L}}{\partial W^{(l)}} = \frac{\partial \mathcal{L}}{\partial h^{(L)}} \cdot \frac{\partial h^{(L)}}{\partial h^{(L-1)}} \cdots \frac{\partial h^{(l+1)}}{\partial h^{(l)}} \cdot \frac{\partial h^{(l)}}{\partial W^{(l)}}$$

Định nghĩa $\delta^{(l)} = \frac{\partial \mathcal{L}}{\partial z^{(l)}}$ (error signal tại lớp l):

$$\delta^{(l)} = \left(W^{(l+1)}\right)^T \delta^{(l+1)} \odot \sigma'\left(z^{(l)}\right)$$

$$\frac{\partial \mathcal{L}}{\partial W^{(l)}} = \delta^{(l)} \left(h^{(l-1)}\right)^T$$

Tổng chi phí: $O(L \cdot n^2)$ per sample: tuyến tính theo số layers, không cần tính lại từ đầu. Đây là lý do deep learning khả thi.

4. Attention Mechanism: Transformer

Scaled Dot Product Attention

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$Q = XW_Q, K = XW_K, V = XW_V$ với $X \in \mathbb{R}^{n \times d}$ (sequence).

Ý nghĩa toán học: $\frac{QK^T}{\sqrt{d_k}}$ tính ma trận similarity $\in \mathbb{R}^{n \times n}$. Sau softmax, mỗi token i nhận weighted sum của tất cả V:

$$\text{output}_i = \sum_{j=1}^{n} \underbrace{\frac{e^{\langle q_i, k_j \rangle / \sqrt{d_k}}}{\sum_l e^{\langle q_i, k_l \rangle / \sqrt{d_k}}}}_{\text{attention weight}} \cdot v_j$$

Chia $\sqrt{d_k}$: tránh dot product quá lớn làm softmax bão hoà (gradient $\to 0$). Complexity: $O(n^2 d)$: tắc nghẽn chính của LLM.

VII. Heuristic và tri thức kinh nghiệm

Phần này trình bày heuristic như một cách đưa tri thức lĩnh vực vào quá trình tìm kiếm. Heuristic không luôn bảo đảm tối ưu toàn cục, nhưng giúp giảm đáng kể không gian cần khảo sát. Đây là lựa chọn thực tế khi bài toán quá lớn, quá nhiễu hoặc quá phức tạp để giải bằng phương pháp chính xác.

Heuristic là chiến lược tìm kiếm dựa trên tri thức lĩnh vực (domain knowledge). Mục tiêu là cắt giảm không gian tìm kiếm và thu được nghiệm tốt trong thời gian thực tế. Heuristic không phải một thuật toán đơn lẻ. Nó là một nhóm phương pháp nằm giữa deterministic, stochastic và approximation.

Định nghĩa hình thức Hàm heuristic $h: \mathcal{S} \to \mathbb{R}^+$ ước lượng chi phí từ trạng thái s đến đích. Thuật toán heuristic dùng h để ưu tiên thứ tự khám phá không gian trạng thái $\mathcal{S}$ thay vì duyệt toàn bộ.

- Constructive Heuristic: Xây dựng nghiệm từ đầu, từng bước chọn lựa tham lam. Greedy, Nearest Neighbor TSP.
- Informed Search: Tìm kiếm có hướng dẫn bởi h. A*, IDA*, RBFS: đảm bảo tối ưu nếu h admissible.
- Local Search: Cải thiện nghiệm hiện tại bằng di chuyển trong lân cận. Hill Climbing, 2 opt, Tabu Search.
- Metaheuristic: Framework cấp cao điều phối local search / population. SA, GA, PSO, ACO, DE.

Nhóm 1: Constructive Heuristic (Deterministic)

Greedy Algorithm: Tổng quát

Tại mỗi bước, chọn lựa tốt nhất cục bộ theo hàm lựa chọn $c: \mathcal{C} \to \mathbb{R}$:

Greedy Template

$$x_k = \operatorname{argmax}_{e \in \mathcal{C}_k} c(e), \quad \mathcal{C}_{k+1} = \mathcal{C}_k \backslash \{x_k\} \cap \text{Feasible}$$

Ví dụ: Huffman Coding: Greedy chọn 2 node tần suất thấp nhất để merge. Chứng minh tối ưu bằng exchange argument:

$$\text{Cost}(T) = \sum_i f_i \cdot d_T(i), \quad f_i = \text{tần suất},\ d_T = \text{độ sâu}$$

Nếu $f_i \le f_j$ thì $d_T(i) \ge d_T(j)$ trong cây tối ưu dẫn đến Greedy đúng vì swap không cải thiện được.

Ví dụ: Nearest Neighbor TSP: Bắt đầu từ thành phố bất kỳ, luôn đến thành phố gần nhất chưa thăm:

$$v_{k+1} = \arg\min_{v \in V \setminus \{v_1, \ldots, v_k\}} d(v_k, v)$$

Thực nghiệm: cho kết quả trong 10 đến 25% OPT, chạy trong $O(n^2)$. Không có bound lý thuyết tốt: worst case tùy ý xấu hơn OPT.

Nhóm 2: Informed Search: Thuật toán A*

A* là heuristic search hoàn chỉnh và tối ưu nếu heuristic thoả điều kiện admissible: cầu nối giữa deterministic (chính xác) và heuristic (nhanh).

Hàm đánh giá A*

$$f(n) = g(n) + h(n)$$

$g(n)$ = chi phí thực từ stkinh nghiệm triển khai đến n; $h(n)$ = ước lượng chi phí từ n đến goal; $h^*(n)$ = chi phí thực tế tối ưu.

Điều kiện Admissible và Consistent

Admissible: $\forall n$: $h(n) \leq h^*(n)$: không được ước lượng quá cao (optimistic).

Consistent (Monotone): $\forall n, n'$: $h(n) \leq c(n, n') + h(n')$: bất đẳng thức tam giác trên h.

Consistent ⇒ Admissible. Nếu h consistent: A* tối ưu và không mở node nào hai lần.

Chứng minh tối ưu của A* (admissible, tree search)

Giả sử A* trả về đường đi p với chi phí $C > C^*$ (OPT). Gọi p^* là đường tối ưu, n là node trên p^* đang chờ trong Open. Tại thời điểm A* chọn goal node:

$$f(n) = g(n) + h(n) \leq g(n) + h^*(n) = C^* < C = f(\text{goal})$$

A* chọn node có f nhỏ nhất dẫn đến A* phải chọn n trước goal dẫn đến mâu thuẫn. ▫

Complexity: Worst case $O(b^d)$ với b = branching factor, d = depth. Chất lượng heuristic quyết định constant: $h \equiv 0$ = BFS; $h = h^* = O(d)$.

Ví dụ: Manhattan Distance cho 8 puzzle

h_Manhattan ≤ h* (Admissible)

$$h_{\text{Man}}(s) = \sum_{i=1}^{8} (|r_i - r_i^*| + |c_i - c_i^*|)$$

Là admissible vì mỗi tile cần ít nhất Manhattan distance bước để về đúng vị trí (mỗi bước chỉ di chuyển 1 ô). Thực nghiệm: h_{Man} mở ít hơn $h \equiv 0$ tới 10^4 lần trên puzzles khó.

Nhóm 3: Local Search và Tabu Search

Hill Climbing và Vấn Đề Local Optima

Hill Climbing: Steepest Ascent

$$s_{t+1} = \arg\max_{s' \in \mathcal{N}(s_t)} f(s'), \quad \text{dừng nếu } f(s_{t+1}) \leq f(s_t)$$

$\mathcal{N}(s)$ = tập lân cận của trạng thái s. Vấn đề: dễ mắc kẹt tại local optimum s^{loc} với $f(s^{\text{loc}}) < f(s^*)$.

Ví dụ TSP với 2 opt: $\mathcal{N}(s)$ = tập tour thu được bằng đổi chỗ 2 cạnh. Mỗi bước: $O(n^2)$ neighbors kiểm tra.

Tabu Search: Bộ Nhớ Ngắn Hạn

Giải quyết local optima bằng cách cấm quay lại trạng thái đã thăm gần đây: biến deterministic local search thành có khả năng thoát local optima.

Tabu Search Framework

$$s_{t+1} = \arg\max_{s' \in \mathcal{N}(s_t) \setminus \mathcal{T}_t} f(s')$$

$\mathcal{T}_t$ = Tabu List kích thước $|\mathcal{T}| = \tau$ lưu τ moves/trạng thái gần nhất bị cấm.

Aspiration Criterion: Nếu move tabu nhưng cho nghiệm tốt hơn best so far, vẫn được chấp nhận:

$$\text{chấp nhận } s' \in \mathcal{T}_t \text{ nếu } f(s') > f(s^{\text{best}})$$

Chọn τ: τ nhỏ dẫn đến dễ cycle; τ lớn dẫn đến không khám phá đủ. Thực nghiệm: $\tau \in [\sqrt{n}, n/5]$ thường tốt. Không có lý thuyết hội tụ mạnh: đây là kinh nghiệm triển khai của Tabu.

Nhóm 4: Evolutionary Algorithms: Genetic Algorithm

Lấy cảm hứng từ tiến hoá Darwin: duy trì quần thể nghiệm, áp dụng selection, crossover, mutation để hội tụ về vùng tốt của không gian tìm kiếm.

GA Framework: Một thế hệ

$$P_{t+1} = \text{Mutation}\Big(\text{Crossover}\big(\text{Selection}(P_t)\big)\Big)$$

Fitness proportional selection (Roulette Wheel):

$$P(\text{chọn } x_i) = \frac{f(x_i)}{\sum_{j=1}^{\mu} f\left(x_j\right)}$$

1 point crossover: Chọn điểm cắt $k \sim \text{Uniform}\{1, \dots, \ell - 1\}$:

$$\text{Offspring}_1 = x_1[1{:}k] \oplus x_2[k+1{:}\ell], \quad \text{Offspring}_2 = x_2[1{:}k] \oplus x_1[k+1{:}\ell]$$

Bit flip mutation: Mỗi bit lật độc lập với xác suất $p_m \approx 1/\ell$:

$$x_i' = x_i \oplus B_i, \quad B_i \sim \text{Bernoulli}(p_m)$$

Schema Theorem (Holland 1975)

Một schema H là template với ký tự $\{0,1,*\}$. Định nghĩa:

$o(H)$ = order (số bit cố định); $\delta(H)$ = defining length (khoảng cách giữa 2 bit cố định ngoài cùng)

$$\mathbb{E}[m(H, t+1)] \geq m(H,t) \cdot \frac{\hat{f}(H,t)}{\bar{f}(t)} \cdot \left(1 - p_c \frac{\delta(H)}{\ell - 1}\right) \cdot (1 - p_m)^{o(H)}$$

Schema ngắn (δ nhỏ), bậc thấp (o nhỏ), fitness cao ($\hat{f} > \bar{f}$) sẽ tăng số lượng trong quần thể: Building Block Hypothesis: GA kết hợp các khối cấu trúc tốt để tạo nghiệm tốt hơn.

Giới hạn toán học của GA Schema Theorem chỉ là lower bound kỳ vọng: không đảm bảo hội tụ toàn cục trong thời gian đa thức. Không có thuật toán tốt nhất cho mọi bài toán: GA không tốt hơn random search trung bình trên mọi bài toán. GA hiệu quả khi bài toán có cấu trúc decomposable: fitness phụ thuộc vào các block con độc lập.

Nhóm 5: Swarm Intelligence: PSO và ACO

Particle Swarm Optimization (PSO)

Mô phỏng hành vi bầy đàn: mỗi hạt di chuyển trong không gian liên tục, bị thu hút bởi vị trí tốt nhất của bản thân và của cả bầy.

PSO: Velocity và Position Update

$$\mathbf{v}_i^{(t+1)} = \underbrace{\omega \mathbf{v}_i^{(t)}}_{\text{inertia}} + \underbrace{c_1 r_1 \odot \left(\mathbf{p}_i - \mathbf{x}_i^{(t)}\right)}_{\text{cognitive}} + \underbrace{c_2 r_2 \odot \left(\mathbf{g} - \mathbf{x}_i^{(t)}\right)}_{\text{social}}$$

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + \mathbf{v}_i^{(t+1)}$$

ω = inertia weight (giảm dần: $0.9 \to 0.4$); $c_1, c_2 \approx 2.0$ = cognitive/social coefficients; $r_1, r_2 \sim$ Uniform$(0,1)$; $\mathbf{p}_i$ = personal best; $\mathbf{g}$ = global best.

Điều kiện hội tụ (Clerc và Kennedy 2002): Hệ thống PSO ổn định khi:

$$\omega < 1, \quad 0 < c_1 + c_2 < 4, \quad \omega > \frac{c_1 + c_2}{2} - 1$$

Hoặc dùng constriction factor $\chi = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|}$ với $\varphi = c_1 + c_2 > 4$.

Ant Colony Optimization (ACO)

Mô phỏng kiến tìm đường: pheromone trail tích luỹ trên những đường đi tốt, tạo cơ chế reinforcement learning dạng stigmergy.

ACO: Xác suất chọn cạnh và Cập nhật Pheromone

Xác suất kiến k đi từ i đến j:

$$p_{ij}^k(t) = \frac{\left[\tau_{ij}(t)\right]^\alpha \cdot \left[\eta_{ij}\right]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta}, \quad j \in \mathcal{N}_i^k$$

τ_{ij} = pheromone; $\eta_{ij} = 1/d_{ij}$ = visibility (heuristic); α, β = trọng số tương đối.

Cập nhật pheromone (ACS: Ant Colony System):

$$\tau_{ij}(t+1) = (1-\rho)\,\tau_{ij}(t) + \sum_{k:(i,j)\in L^k} \frac{1}{C^k}$$

$\rho \in (0,1)$ = evaporation rate; C^k = độ dài tour của kiến k. Evaporation tránh stagnation: pheromone cũ quên dần.

Phân tích toán học: ACO tương đương với Reinforcement Learning dạng tabular với reward $\propto 1/C^k$ và temporal discount. Hội tụ đến tối ưu toàn cục chứng minh được khi $t \to \infty$ (Dorigo và Blum 2005), nhưng tốc độ hội tụ chậm hơn SA về mặt lý thuyết.

Nhóm 6: Hybrid Heuristics: GRASP và Memetic

GRASP (Greedy Randomized Adaptive Search Procedure)

Mỗi iteration: (1) Construction phase với Restricted Candidate List (RCL):

$$\text{RCL} = \{e \in \mathcal{C}: c(e) \geq c_{\min} + \alpha(c_{\max} - c_{\min})\}, \quad \alpha \in [0,1]$$

Chọn ngẫu nhiên từ RCL thay vì greedy tuyệt đối. (2) Local search phase cải thiện nghiệm xây được.

$\alpha = 0$: GRASP = Greedy (deterministic); $\alpha = 1$: GRASP = random construction. $\alpha \in [0.2,0.5]$ thường tốt nhất.

Memetic Algorithm: GA + Local Search

$$P_{t+1} = \text{LocalSearch}\big(\text{GA\_Generation}(P_t)\big)$$

Sau mỗi crossover/mutation, áp dụng local search (2 opt, 3 opt, hill climbing) lên từng cá thể trước khi đánh giá fitness. Lamarckian evolution: học kinh nghiệm trong đời (local optima) truyền cho thế hệ sau.

Hiệu quả: Memetic giảm số thế hệ cần thiết 10 đến 100 × so với plain GA nhờ local refinement, đặc biệt trong TSP, VRP, scheduling.

Phân loại tổng hợp: Heuristic Taxonomy

Thuật toán	**Nhóm chính**	**Xác định/Ngẫu nhiên**	**Bảo đảm tối ưu**	**Complexity**	**Bài toán phù hợp**
Greedy	Constructive	Deterministic	Không (trừ special cases)	O(n log n)	Huffman, MST, Scheduling
A* / IDA*	Informed Search	Deterministic	Nếu h admissible	O(b^d)	Pathfinding, Puzzle, Planning
Hill Climbing	Local Search	Deterministic	Local optimum only	O(n; |N|)	Continuous optimisation, SAT
2 opt / 3 opt	Local Search	Deterministic	Local 2 opt/3 opt optimal	$O(n^2)$ đến $O(n^3)$	TSP, VRP
Tabu Search	Local Search + Memory	Deterministic*	Không (thực nghiệm tốt)	O(iter; |N|)	Job Shop, QAP, TSP
Simulated	Metaheuristic	Stochastic	Prob. global	O(iter)	Continuous,

Annealing			optimal (log schedule)		discrete, VLSI layout
Genetic Algorithm	Evolutionary	Stochastic	Không chắc	O(μ; ℓ; gen)	Multi objective, black box opt
PSO	Swarm Intelligence	Stochastic	Không chắc	O(swarm; iter; d)	Continuous optimisation, NN training
ACO	Swarm Intelligence	Stochastic	Asymptotic global	O(ant; n^2; iter)	TSP, routing, scheduling
GRASP	Hybrid Constructive	Stochastic	Không	O(iter; (n + LS))	Set Cover, Max Cut, VRP
Memetic (MA)	Evolutionary + LS	Stochastic	Không (thực nghiệm rất tốt)	O(μ; LS; gen)	TSP, protein folding, scheduling

Thách thức chung của Heuristic

Parameter tuning: SA cần lịch làm nguội; GA cần μ, p_c, p_m; PSO cần ω, c_1, c_2. Không có lý thuyết tổng quát chọn tham số tối ưu: thường cần meta optimization (grid search, Bayesian optimization).

No convergence guarantee: Trừ SA với log schedule và A* với admissible h, phần lớn metaheuristic không có bound hội tụ thực tế. Kết quả phụ thuộc vào random seed, budget (iterations).

Problem specific tuning: ACO cho TSP khác ACO cho scheduling. Việc điều chỉnh $\mathcal{N}(s)$, encoding, operators cho từng bài toán đòi hỏi domain expertise sâu: đây là kinh nghiệm thiết kế của metaheuristic.

VIII. Vì sao AI có hiệu quả dưới góc nhìn toán học?

Phần này tổng hợp các lý do toán học giải thích hiệu quả của AI hiện đại. Trọng tâm không nằm ở việc xem AI như một cơ chế đặc biệt, mà ở cách dữ liệu thực tế có cấu trúc, cách mô hình tận dụng cấu trúc đó và cách optimization tìm được nghiệm có khả năng tổng quát hóa.

Hiệu quả của AI không xuất phát từ một cơ chế đặc biệt duy nhất. Có ít nhất sáu lý do toán học cần được xem xét: cấu trúc thấp chiều của dữ liệu, inductive bias, regularization ngầm của SGD, overparameterization, transfer learning và sự tồn tại của sparse subnetworks.

Lý do 1: Phá vỡ Curse of Dimensionality (một phần)

Manifold Hypothesis

Dữ liệu thực tế $\mathbf{x} \in \mathbb{R}^D$ (ảnh 224×224×3 dẫn đến $D \approx 150K$) thực sự nằm trên hoặc gần một manifold thấp chiều $\mathcal{M} \subset \mathbb{R}^D$ với $\dim(\mathcal{M}) = d \ll D$.

$$\mathbf{x} \approx g(\mathbf{z}), \quad \mathbf{z} \in \mathbb{R}^d, \quad d \ll D$$

Deep network học encoder $f: \mathbb{R}^D \to \mathbb{R}^d$ và decoder $g: \mathbb{R}^d \to \mathbb{R}^D$. Thay vì làm việc trong $\mathbb{R}^{150K}$, thực sự chỉ cần $\mathbb{R}^d$ với $d \sim 10$ đến 100.

Lý do 2: Inductive Bias = Prior Knowledge

CNN: Translational Equivariance

$$f\big(\tau_t(\mathbf{x})\big) = \tau_t\big(f(\mathbf{x})\big) \quad \forall \text{ translation } \tau_t$$

CNN không cần học riêng đối tượng ở góc trái và đối tượng ở góc phải: cùng kernel áp dụng mọi nơi. Giảm số tham số từ $O(H \cdot W \cdot C_{in} \cdot C_{out})$ xuống $O(k^2 \cdot C_{in} \cdot C_{out})$ (shared weights).

Graph Network: f bất biến hoặc đồng biến với hoán vị node: phù hợp dữ liệu đồ thị phân tử, mạng xã hội.

Lý do 3: Implicit Regularization của SGD

SGD tìm nghiệm Flat Minima

Với loss landscape không lồi, có vô số global minima (overparameterized networks). SGD ưu tiên nghiệm có Hessian nhỏ (flat minimum):

$$\text{Sharpness} = \lambda_{\max}\big(\nabla^2 \mathcal{L}(\theta)\big) \quad \text{(nhỏ = flat)}$$

Flat minima generalise tốt hơn (Hochreiter và Schmidhuber 1997; Li et al. 2018): nhiễu nhỏ $\Delta\theta$ thay đổi loss ít hơn ở flat region. SGD với noise tự nhiên $\propto \sigma^2/b$ có xu hướng drift đến flat minima.

Bayesian interpretation: SGD ≈ MAP inference với prior $\propto e^{-\lambda_{\max}/2\sigma^2}$.

Lý do 4: Overparameterization và Double Descent

Bias Variance Decomposition tổng quát

$$\mathbb{E}\left[\left(y-\hat{f}(x)\right)^2\right] = \underbrace{\text{Bias}^2\left[\hat{f}(x)\right]}_{\text{underfitting}} + \underbrace{\text{Var}\left[\hat{f}(x)\right]}_{\text{overfitting}} + \sigma_\varepsilon^2$$

Lý thuyết cổ điển: thêm tham số dẫn đến Var tăng dẫn đến lỗi tăng. Thực tế modern ML:

Double Descent Phenomenon (Belkin et al. 2019) Khi số tham số p vượt qua interpolation threshold $p > n$ (số mẫu), risk giảm trở lại:

$$R(p) \approx \begin{cases} \text{tăng} & p < n \\ \text{peak} & p \approx n \\ \text{giảm} & p \gg n \end{cases}$$

Với $p \to \infty$ và implicit regularisation: $R \to$ giá trị tốt hơn cả underparameterized regime!

Lý do 5: Transfer Learning và Feature Reuse

Tại sao pre training + fine tuning hiệu quả

Gọi $\phi: \mathbb{R}^D \to \mathbb{R}^d$ là feature extractor pre trained. Với task mới chỉ cần học:

$$\min_W \frac{1}{n}\sum_{i=1}^{n} \ell\left(W \cdot \phi(x_i), y_i\right)$$

Thay vì $\min_{\phi,W}$: bài toán non convex $d \times D + d \times C$ chiều. Fine tuning: W tuyến tính dẫn đến bài toán lồi! Mọi local minimum là global minimum.

Bất đẳng thức sample complexity: Fine tuning cần $n_{\text{finetune}} \ll n_{\text{scratch}}$:

$$n_{\text{finetune}} = O\left(\frac{d_{\text{task}}}{\varepsilon^2}\right) \quad \text{vs} \quad n_{\text{scratch}} = O\left(\frac{D}{\varepsilon^2}\right)$$

Tiết kiệm D/d_{task} lần dữ liệu: thường 10^2 đến 10^4 lần.

Lý do 6: Lottery Ticket và Sparse Subnetworks

Lottery Ticket Hypothesis (Frankle và Carlin 2019) Trong mạng dày f_θ (overparameterized), tồn tại subnetwork f_{θ_s} (sparse, $|\theta_s| \ll |\theta|$) với mask m sao cho:

$$f_{\theta \odot m}(\mathbf{x}) \approx f_\theta(\mathbf{x}) \quad \forall \mathbf{x}$$

và khi huấn luyện riêng từ khởi tạo ban đầu $\theta_s^{(0)} = m \odot \theta^{(0)}$, đạt kết quả tương đương.

Ý nghĩa: Overparameterization không dư thừa hoàn toàn: nó tạo ra landscape thuận lợi để SGD tìm được subnetwork tốt. Pruning sau đó giảm 80 đến 90% tham số mà chỉ mất <1% accuracy.

Tổng hợp: So sánh định lượng

Phương pháp	**Bài toán phù hợp**	**Độ phức tạp**	**Bảo đảm**	**Hạn chế chính**
Deterministic	Hệ PT tuyến tính, đồ thị, sắp xếp	Poly	Exact, 100%	NP hard intractable
Stochastic	Tích phân chiều cao, sampling, tối ưu toàn cục	O(1/√N)	Xác suất cao	Variance, mixing time
Approximation	NP hard combinatorial (Vertex Cover, TSP)	Poly	α-optimal	Inapproximability gap
Interpolation	Tái tạo tín hiệu, CAD, surrogate model	O(n) đến $O(n^3)$	Exact tại nút	Runge, noisy data
Heuristic (A*, Greedy)	Informed search, constructive	O(b^d) đến O(n log n)	Optimal nếu h admissible	Parameter tuning, local optima
Metaheuristic (GA, PSO, ACO)	NP hard combinatorial, black box, multi obj	O(pop; iter)	Không bảo đảm	No convergence proof, tuning
ML (Classical)	Classification, regression, clustering	$O(nd^2)$	PAC bounds	Feature engineering
Deep Learning	Vision, NLP, RL, generative modeling	O(LBnd)	Empirical SOTA	Interpretability, data hunger
LLM (Transformer)	Ngôn ngữ, code, reasoning, multi modal	$O(n^2d)$ per token	Emergent capabilities	Hallucination, $O(n^2)$ context

Không có thuật toán tốt nhất cho mọi bài toán Theorem (Wolpert 1996): Không thuật toán nào tốt nhất trên mọi phân phối bài toán:

$$\sum_f P\,(\text{success}|f, \mathcal{A}_1) = \sum_f P\,(\text{success}|f, \mathcal{A}_2) \quad \forall \mathcal{A}_1, \mathcal{A}_2$$

AI/Deep Learning hiệu quả vì thế giới thực không phải phân phối đều: dữ liệu thực có cấu trúc phong phú (manifold thấp chiều, tương quan cục bộ, tính đối xứng). Deep network exploit chính xác cấu trúc đó thông qua inductive bias phù hợp. Nói cách khác: AI có hiệu quả vì dữ liệu thực tế có cấu trúc, không phải vì toán học cơ chế đặc biệt.

Kết luận

Các phương pháp tính toán không tồn tại độc lập theo nghĩa tuyệt đối. Trong thực tế, một hệ thống có thể kết hợp deterministic để xử lý các bước có cấu trúc rõ ràng, stochastic để lấy mẫu hoặc tối ưu trong không gian lớn, approximation để giảm chi phí, interpolation để suy luận từ dữ liệu hữu hạn và heuristic để đưa tri thức lĩnh vực vào quá trình tìm kiếm.

Điểm quan trọng là phải chọn phương pháp theo bản chất của bài toán. Nếu bài toán có nghiệm chính xác và chi phí tính toán chấp nhận được, deterministic là lựa chọn tự nhiên. Nếu bài toán có không gian trạng thái lớn, dữ liệu nhiễu hoặc phân phối phức tạp, stochastic và heuristic thường phù hợp hơn. Nếu bài toán yêu cầu cân bằng giữa độ chính xác và hiệu năng, approximation và optimization giữ vai trò trung tâm.

Machine Learning và AI hiện đại có thể được hiểu như sự kết hợp có hệ thống của các nền tảng trên. Mô hình học máy biểu diễn hàm bằng approximation, cập nhật tham số bằng optimization, xử lý dữ liệu lớn bằng stochastic và tận dụng cấu trúc dữ liệu thông qua inductive bias. Vì vậy, AI không thay thế các phương pháp tính toán cổ điển. AI mở rộng các phương pháp đó trong bối cảnh dữ liệu lớn, mô hình nhiều tham số và yêu cầu suy luận phức tạp.

Khi sử dụng tài liệu này nên đọc mỗi phương pháp theo ba câu hỏi. Bài toán giả định điều gì. Phương pháp bảo đảm được điều gì. Giới hạn tính toán hoặc giới hạn mô hình nằm ở đâu. Ba câu hỏi này giúp người học tránh xem thuật toán như công thức rời rạc và hiểu đúng vai trò của từng phương pháp trong thiết kế hệ thống tính toán.