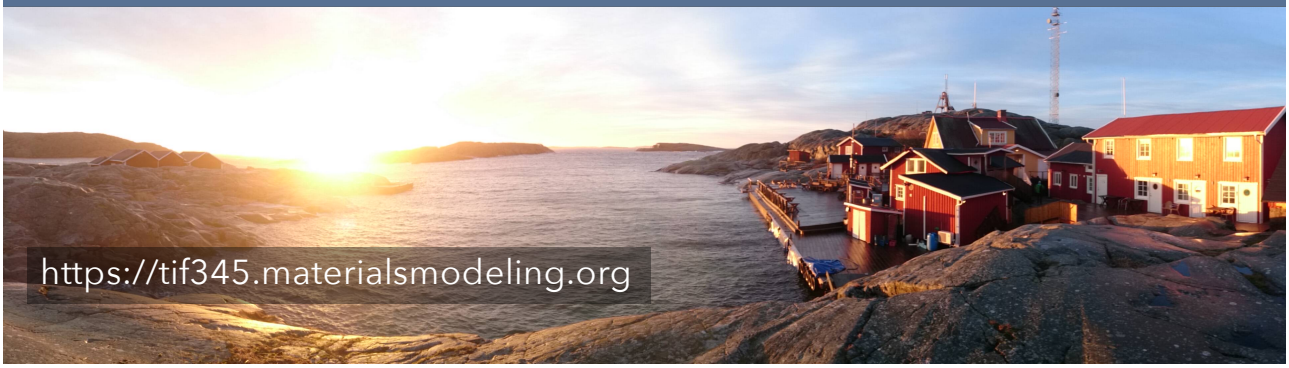
TIF345 / FYM345
Advanced Simulation and Machine Learning

Paul Erhart
Andreas Ekström
Bernhard Mehlig
Arkady Gonoskov



Plan for the next two weeks

1-2 Foundations: Linear models, Gaussian processes

3-4 Applications: Linear models, Gaussian processes, Regression

5-7 Neural networks and other ML techniques

tif/fim{345}
Advanced Simulation and Machine Learning

L1: Cluster expansions (CEs) and using the covariance matrix

L2: Feature selection algorithms

→ P2a: Regression with CEs

L3: Sensitivity analysis

L4: Global optimization

→ P2b: GPs and Bayesian optimization

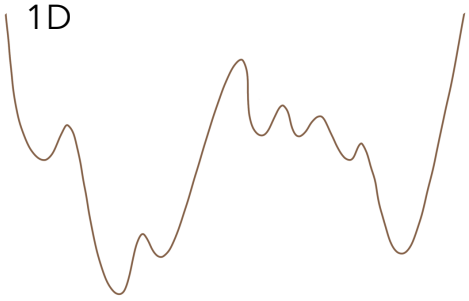
5. Advanced regression
6. Toward applications
   - 6.1. Alloy cluster expansions
   - 6.2. Phonons and force constants
   - 6.3. Interatomic potentials
7. Compressive sensing
8. Sensitivity analysis
9. Global optimization

Demos

Global optimization

Global optimization

1D



2D

Stochastic methods

(Meta)heuristic methods

Possible targets

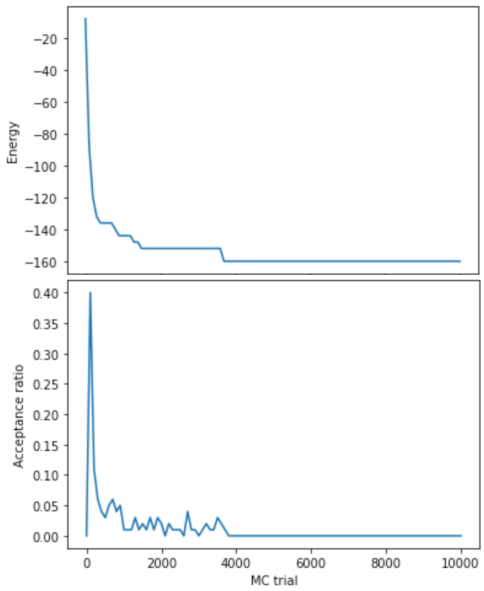
- Atomic structures
- Model loss functions
- Instrument settings
- Path finding
- Network optimization
- …

Response surface methodology-based

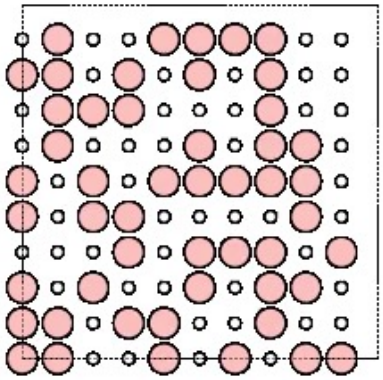
Stochastic methods: Simulated annealing

$T_{initial}$ → T_{final}: What is optimal?

$$T_i = \frac{T_0}{\log(i + i_0)}, i \geq 1$$



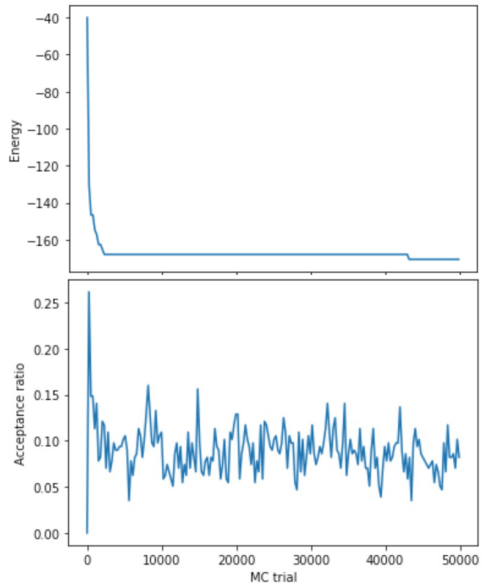
$$S = k_B \ln \Omega$$
$$F = U - TS$$
$$\mathcal{Z} = \sum_i \exp\left[-E_i/k_B T\right]$$

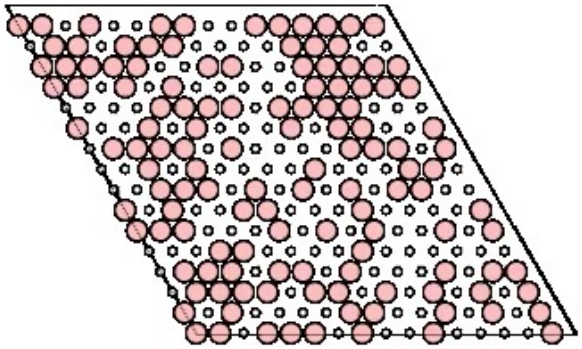


Stochastic methods: Simulated annealing

What causes the difference in behavior?

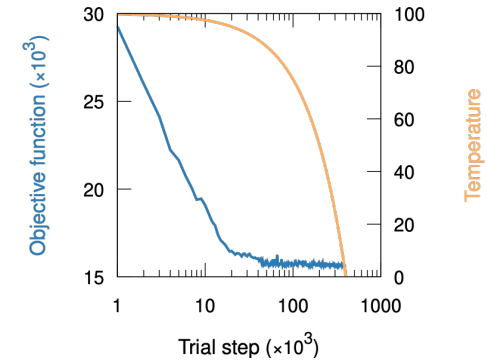
$$S = k_B \ln \Omega$$
$$F = U - TS$$
$$\mathcal{Z} = \sum_i \exp\left[-E_i/k_B T\right]$$

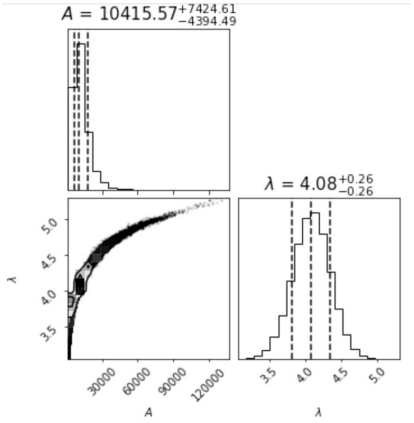


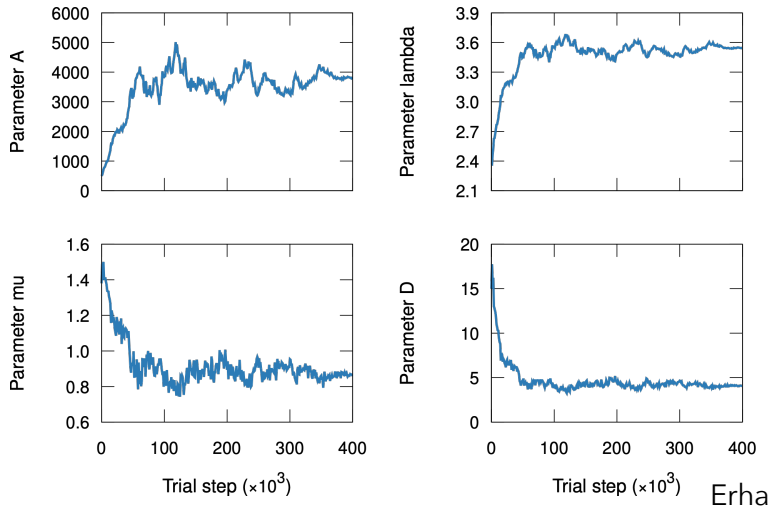


Stochastic methods: Simulated annealing

Optimizing a sloppy model







CHALMERS UNIVERSITY OF TECHNOLOGY

Erhart

Stochastic methods: Critical slowing down

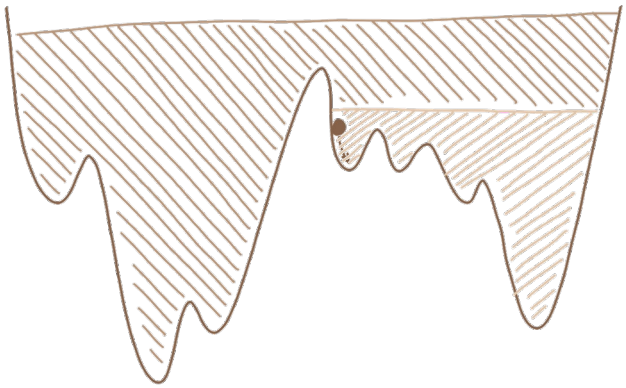
Optimization is probing a *free energy* landscape

$$S = k_B \ln \Omega$$
$$F = U - TS$$

Zero temperature

High temperature

Lower temperature

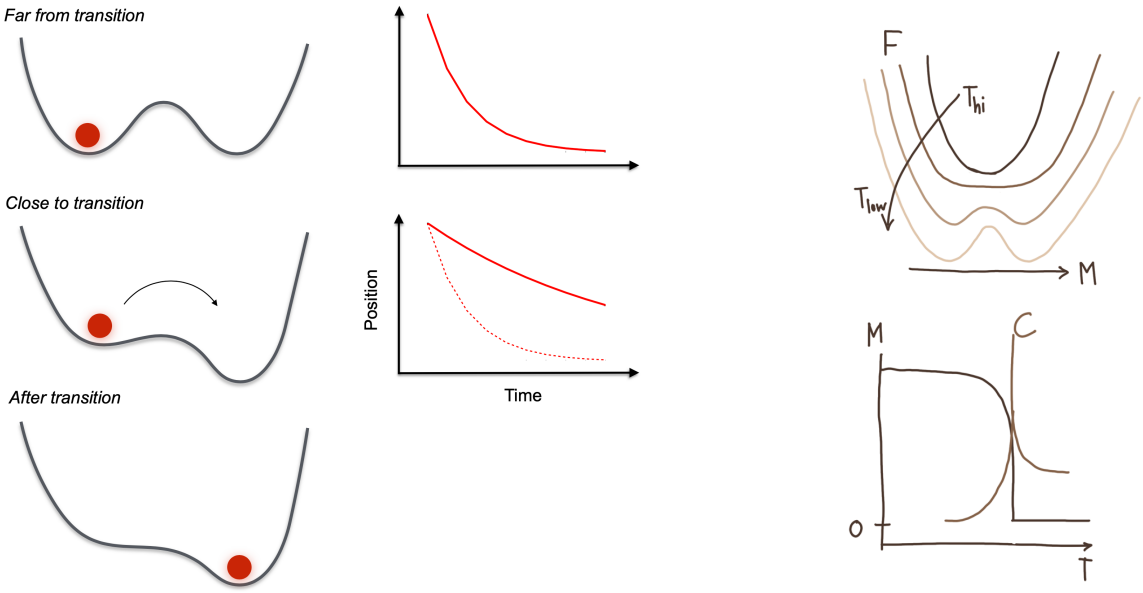


CHALMERS UNIVERSITY OF TECHNOLOGY

Erhart

Stochastic methods: Critical slowing down

Near phase transitions systems recover more slowly from small perturbations



Meisel, *The functional role of critical dynamics in neural systems*, 10.1007/978-3-030-20965-0_4 (2019)

Stochastic methods: Parallel tempering

Parallel tempering (aka Replica exchange) → overcome critical slowdown

N statistically independent replicas
Sampled at temperatures T_i (schedule)
Exchange replicas with probability (satisfying detailed balance)

$$\mathcal{P} = \min\left[1, \frac{\exp\left(-\frac{E_j}{k_B T_i} - \frac{E_i}{k_B T_j}\right)}{\exp\left(-\frac{E_i}{k_B T_j} - \frac{E_j}{k_B T_i}\right)}\right] = \min\left[1, \exp\left((E_i - E_j)\left(\frac{1}{k_B T_i} - \frac{1}{k_B T_j}\right)\right)\right]$$

Acceptance rate at lower *T* where *R*=*T*′/*T*

$$Ac(T,T') \simeq \frac{2}{B(d/2,d/2)} \int_0^{1/(1+R)} \theta^{d/2-1}(1-\theta)^{d/2-1} d\theta$$

How do you choose *R*?

Now: geometric progression

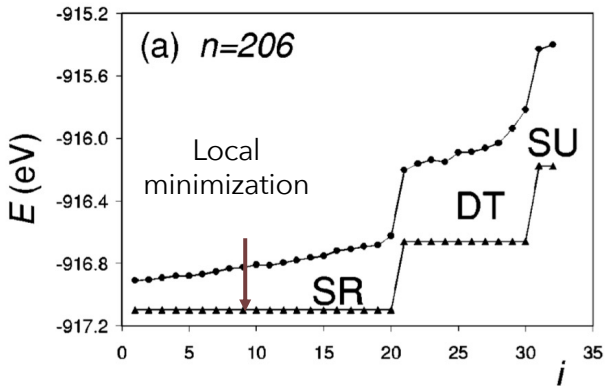
$$T_i = R^{i-1} T_{\min}, 1 \le i \le N$$

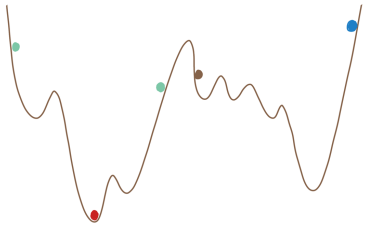
In simulated annealing (log)

$$T_i = \frac{T_0}{\log(i + i_0)}, i \ge 1$$

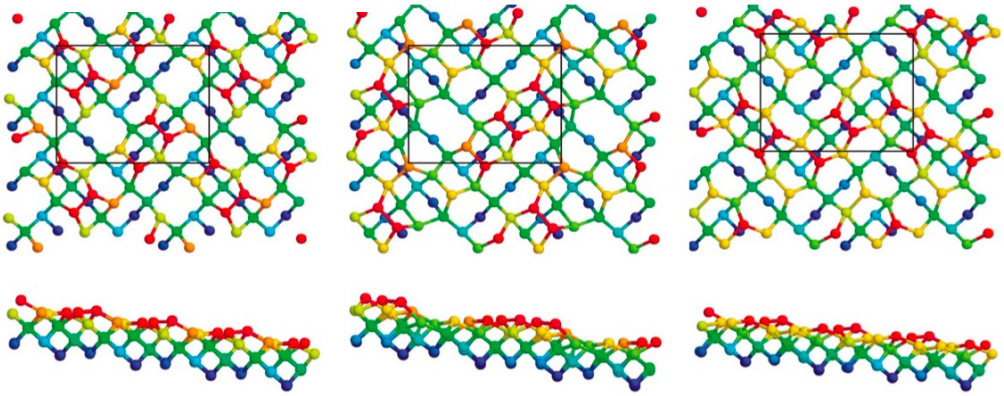
Ciobano and Predescu, Phys. Rev. B **70**, 085321 (2004)

Stochastic methods: Example Si (105) surface





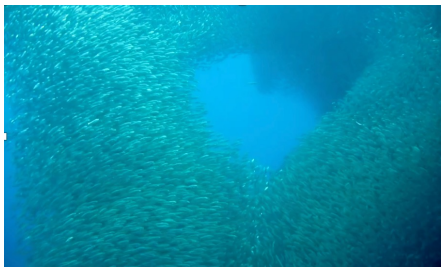
Parallel tempering combined with simulated annealing



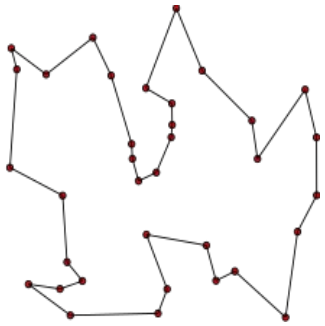
Iddano and Predescu, Phys. Rev. B 70, 085321 (2004)

(Meta)heuristic methods

1. Particle swarm optimization



2. Ant colony optimization



3. Evolutionary algorithms

Particle swarm optimization

https://www.youtube.com/watch?v=Y-5ffl5_7AI
https://www.youtube.com/watch?v=eakKfY5aHmY

Particle swarm optimization: Example algo 1

Consider a swarm of particles, each of which
- has a position and a velocity,
- knows the value of the objective function for its position,
- remembers its best previous position
- knows the best positions and respective functions of its neighbors.

$$\vec{v}_{i,t+1} = c_1\vec{v}_{i,t} + c_2(\vec{\tilde{x}}_{i,t} - \vec{x}_{i,t}) + c_3(\vec{\tilde{x}}_{i-\mathrm{nbr},t} - \vec{x}_{i,t})$$
$$\vec{x}_{i,t+1} = \vec{x}_{i,t} + \vec{v}_{i,t+1},$$

c_1, c_2, c_3: social/cognitive confidence coefficients
- c_1 quantifies how much the particle trusts *itself* now
- c_2 measures how much the particle trusts its experience
- c_3 determines how much the particle trusts its *neighbors*

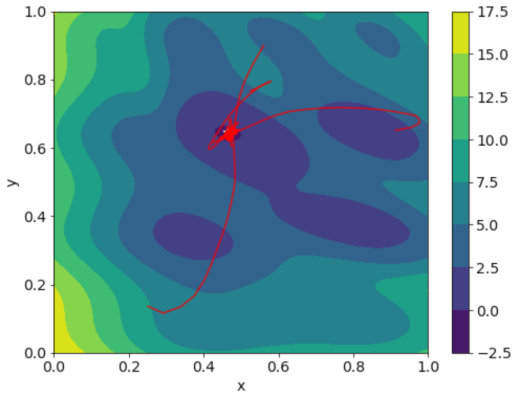
Particle swarm optimization: Example algo 2

Consider a swarm of particles, each of which

1. Evaluate the objective function, yielding the value u_j for particle j
   - If u_j is lower than the previous lowest value for this particle, at position $\mathbf{p_j}$, overwrite $\mathbf{p_j}$
   - If uj is lower than the previous lowest value for all particles, at position $\mathbf{g}$, overwrite $\mathbf{g}$ (global best position)
2. Update the velocity, $\mathbf{v_{new}}$
   $\mathbf{v_{new}} = \omega\mathbf{v} + c_1 R_1(\mathbf{p_j}-\mathbf{x}) + c_2 R_2(\mathbf{g}-\mathbf{x})$,
3. Update the position, $\mathbf{x_{new}}=\mathbf{x}+\mathbf{v_{new}}\Delta t$

- $c_1 \rightarrow \infty$: exploration
- $c_2 \rightarrow \infty$: exploitation

- ω, R_1, R_2: random numbers



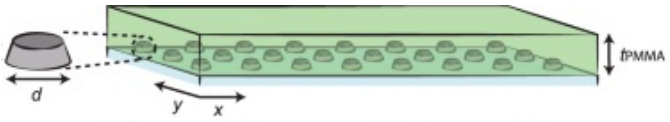
Demo

Images from https://wikipedia.org

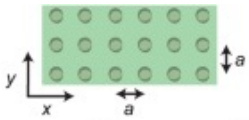
PSO for optimizing hydrogen sensors

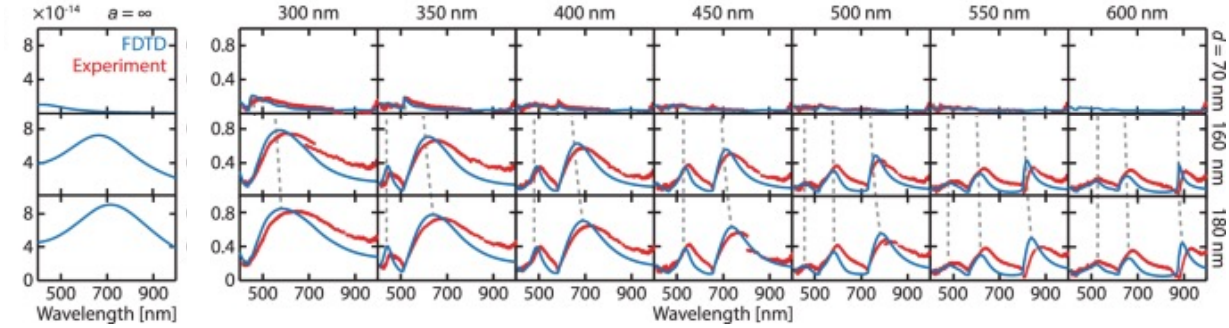
Pd nanoparticles for H_2 detection

Random array



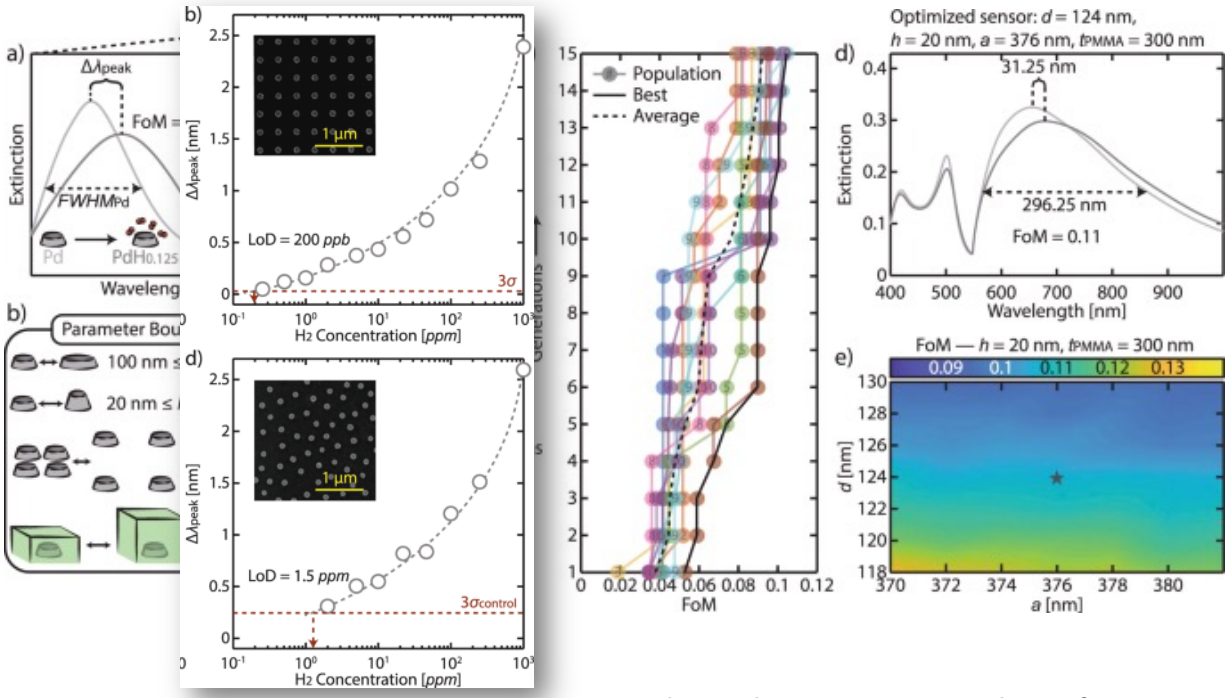
Ordered array





Nugroho *et al.*, Nature Comm. **13**, 5737 (2022)

PSO for designing nanoparticle arrays



Final result (experimental verification!)

Nugroho *et al.*, Nature Comm. **13**, 5737 (2022)

Natural evolution strategies (NES)

Evolutionary algorithms optimize by "fitness"
Natural ESs operate on multivariate Gaussian mutation distribution
→ Only function (no gradient) evaluations still comparable to 2nd order

What does it do?
It evolves a "search distribution"
by following a "sampled natural gradient"

→ Assume each parameter is given by a (normal) distribution
→ The "search distribution" is defined by $\theta = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$
with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
→ The gradient of the expected "fitness" can be sampled
using the "log-likelihood" trick

Glasmachers *et al.*, GECCO '10 doi:10.1145/1830483.1830557 (2010)

The log-likelihood trick

Density of normal search distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \rightarrow \mathcal{N}(\mathbf{0}, \mathbf{1})$

$$p(x \mid \theta) = \frac{1}{(\sqrt{2\pi})^d \det(A)} \cdot \exp\left(-\frac{1}{2}\left\|A^{-1} \cdot (x - \mu)\right\|^2\right) \quad \text{with} \quad \mathbf{A}\mathbf{A}^T = \boldsymbol{\Sigma}$$

Expected fitness under search distribution [$f(x)$ = fitness measure]

$$J(\theta) = \mathbb{E}[f(x) \mid \theta] = \int f(x)\, p(x \mid \theta)\, dx$$

$$\begin{aligned}\nabla_\theta J(\theta) &= \nabla_\theta \int f(x)\, p(x \mid \theta)\, dx \\ &= \int f(x)\, \nabla_\theta p(x \mid \theta)\, dx \\ &= \int f(x)\, \nabla_\theta p(x \mid \theta)\, \frac{p(x \mid \theta)}{p(x \mid \theta)}\, dx\end{aligned}$$

Sample via MC (derivative can be computed efficiently)

$$\nabla_\theta J(\theta) \approx \frac{1}{n} \sum_{i=1}^{n} f(x_i)\, \nabla_\theta \log(p(x \mid \theta))$$

Glasmachers *et al.*, GECCO '10 doi:10.1145/1830483.1830557 (2010)

Further improvements

Use "natural" gradient

$$G = F^{-1} \nabla_\theta J(\theta)$$

F: Fisher information metric

Fitness shaping: fitness → utility

$$\nabla_\theta J(\theta) \approx \frac{1}{n} \sum_{i=1}^{n} f(x_i)\, \nabla_\theta \log(p(x \mid \theta))$$

$$\rightarrow \quad \nabla_\theta \widehat{J}(\theta) = \sum_{i=1}^{n} u_i\, \nabla_\theta \log(p(x_i \mid \theta))$$

$u_1 \geq \ldots \geq u_n$: ranked utility values for each member of population

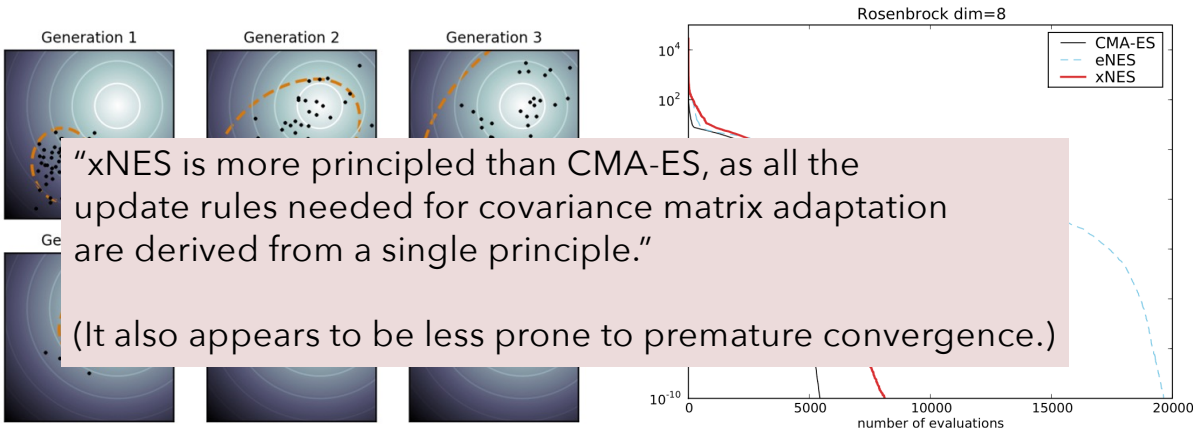
Glasmachers *et al.*, GECCO '10 doi:10.1145/1830483.1830557 (2010)

Further improvements

eNES = exact NES
- Improved handling of Fisher information metric
- Introduces important mixing

xNES = exponential NES
- Exponential space instead of directly on **Σ** to ensure positive-definite **Σ**)
- Avoids computation of Fisher metric by transforming coordinate system

Covariance matrix adaptation evolution strategy (CMA-ES)



"xNES is more principled than CMA-ES, as all the update rules needed for covariance matrix adaptation are derived from a single principle."

(It also appears to be less prone to premature convergence.)

Glasmachers *et al.*, GECCO '10 doi:10.1145/1830483.1830557 (2010); wikicommons

xNES by example

Task: Training a neural network potential (beyond SGD)

$N_{\mathrm{par}} = (N_{\mathrm{des}} + 2)N_{\mathrm{neu}} + 1$ ~ typically a few thousand

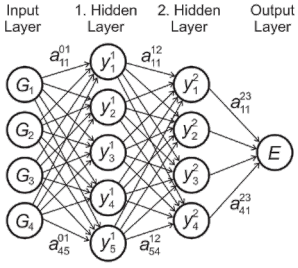
N_{des}: Number of nodes in input (descriptor) layer
N_{neu}: Number of neurons in hidden layer

Loss function

$$L(\mathbf{z}) = \lambda_{\mathrm{e}} L_{\mathrm{e}}(\mathbf{z}) + \lambda_{\mathrm{f}} L_{\mathrm{f}}(\mathbf{z}) + \lambda_{\mathrm{v}} L_{\mathrm{v}}(\mathbf{z}) + \lambda_1 L_1(\mathbf{z}) + \lambda_2 L_2(\mathbf{z})$$

energies forces virials ℓ_1 ℓ_2

z: parameter vector



Fan *et al.*, Phys. Rev. B **104**, 104309 (2021)

xNES by example

1. Initialization

$\mathbf{m}_{\text{ini}} \leftarrow \text{rand}\left[-\frac{1}{2}, \frac{1}{2}\right]$ Mean vector

$\mathbf{s}_{\text{ini}} = 0.1$ Standard deviation

$\mathbf{m}, \mathbf{s} \in \mathbb{R}^{N_{\text{par}}}$

2. Loop over generations
   1. Create a population of solutions $\mathbf{z}_k \ (1 \leqslant k \leqslant N_{\text{pop}})$

$$\mathbf{z}_k \leftarrow \mathbf{m} + \mathbf{s} \odot \mathbf{r}_k \qquad \mathbf{r}_k \sim \mathcal{N}(0, 1)$$

   2. Evaluate loss function and sort populations from small to large
   3. Update natural gradients (u_i rank-based utility values)

$$\nabla_{\mathbf{m}} J \leftarrow \sum_{k=1}^{N_{\text{pop}}} u_k \mathbf{r}_k,$$

   4. Update mean and sdv

$$\mathbf{m} \leftarrow \mathbf{m} + \eta_{\mathbf{m}} (\mathbf{s} \odot \nabla_{\mathbf{m}} J)$$

$$\nabla_{\mathbf{s}} J \leftarrow \sum_{k=1}^{N_{\text{pop}}} u_k (\mathbf{r}_k \odot \mathbf{r}_k - 1)$$

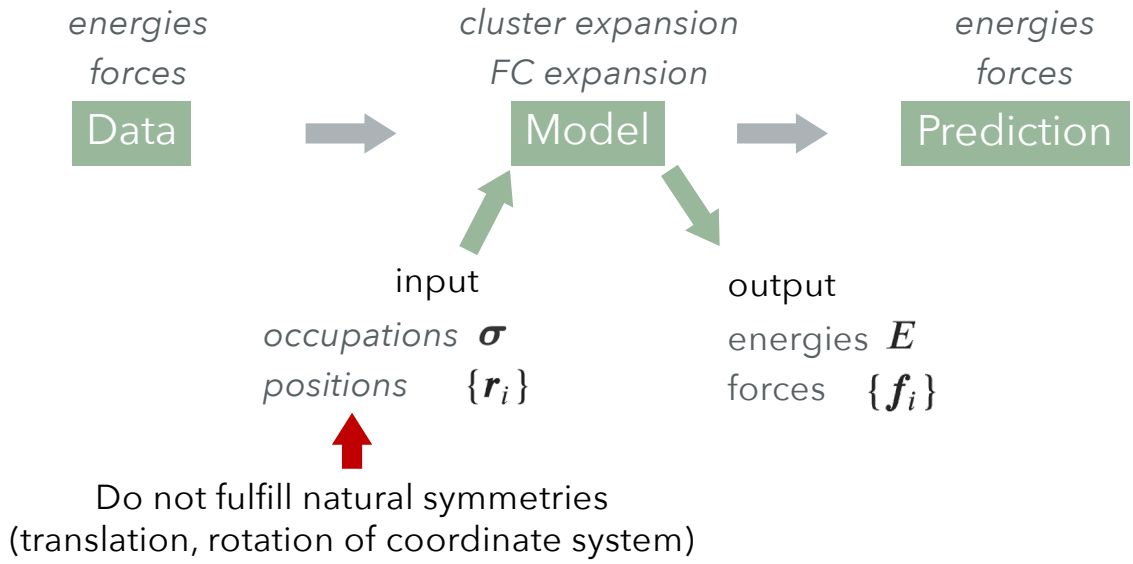
$$\mathbf{s} \leftarrow \mathbf{s} \odot \exp\left(\frac{\eta_{\mathbf{s}}}{2} \nabla_{\mathbf{s}} J\right),$$

$\eta_{\mathbf{s}}$, $\eta_{\mathbf{m}}$: "learning rates"

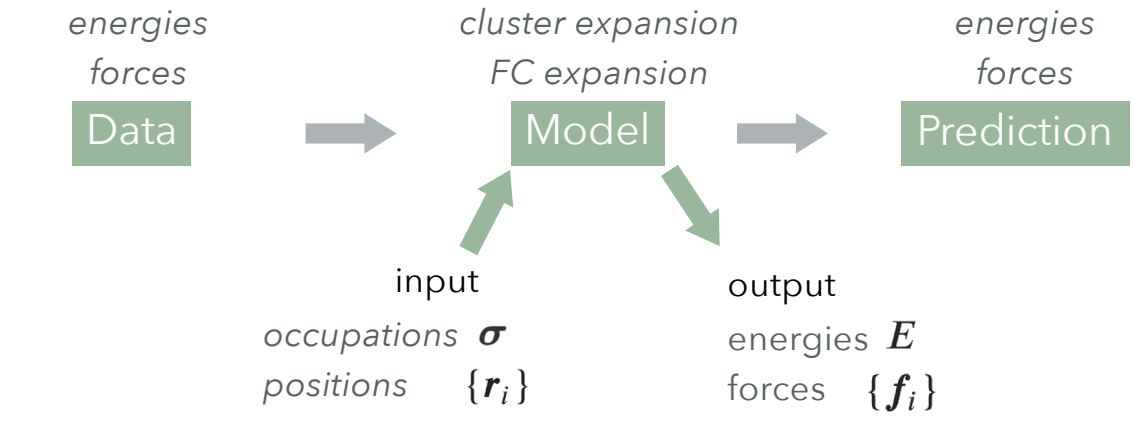
Glasmachers *et al.*, GECCO '10 doi:10.1145/1830483.1830557

Injection:
Machine learned interatomic potentials

How do we represent the state of a system?



How do we represent the state of a system?



Consider a cluster expansion

$$E(\boldsymbol{\sigma}) = E_0 + \sum_i J_i \sigma_i + \sum_{ij} J_{ij} \sigma_i \sigma_j + \sum_{ijk} J_{ijk} \sigma_i \sigma_j \sigma_k + \dots$$

$$E_{\mathrm{CE}} = J_0 + \sum_\alpha m_\alpha J_\alpha \Pi_\alpha(\boldsymbol{\sigma})$$

Coordinate transformation

How do we represent the state of a system?

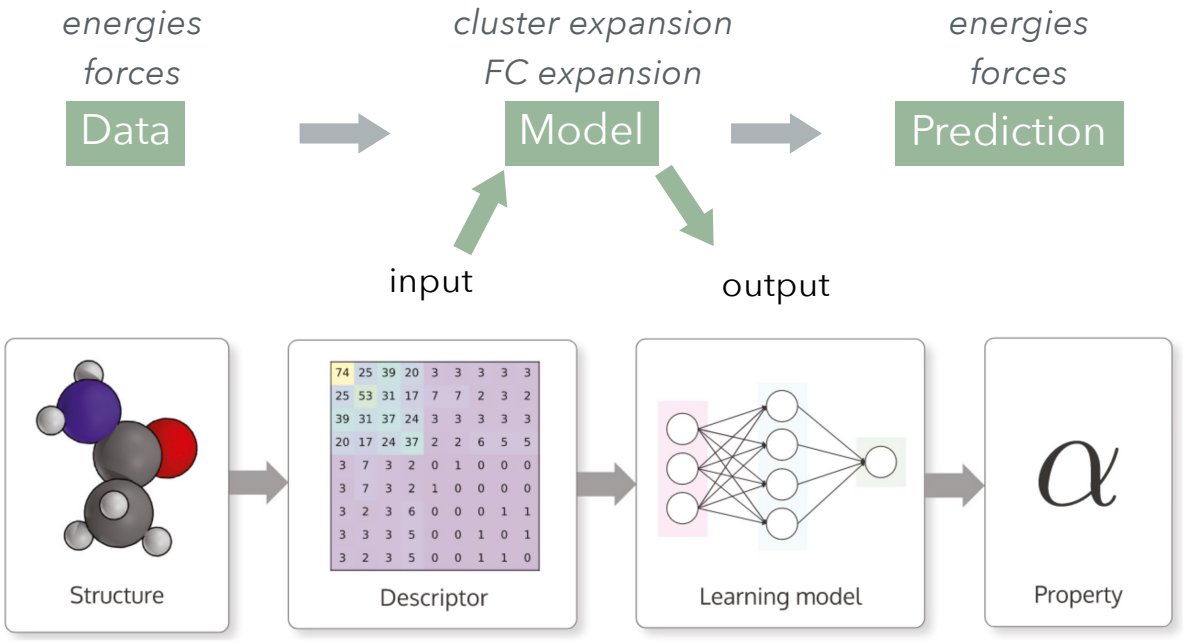


Figure from Comp. Phys. Comm. **247**, 106949 (2020)

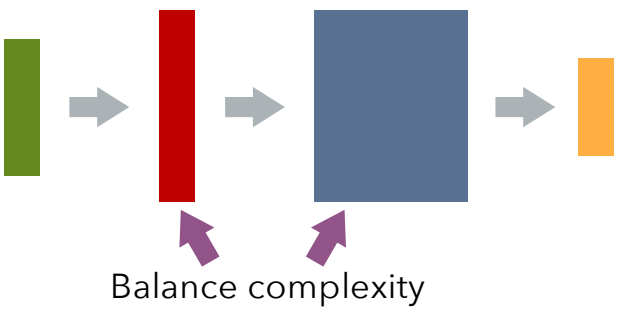
Model vs descriptor complexity

transformation

CE $\boldsymbol{\sigma} \Rightarrow \Pi_\alpha = \xi_k$

FC $\{\boldsymbol{r}_i\} \Rightarrow \{\boldsymbol{u}_i\} \Rightarrow \left\langle u_i^\alpha u_j^\beta \ldots \right\rangle$

Linear models → MBTP
Kernel trick → sGDML
Gaussian process → GAP
Neural network → NNP
…



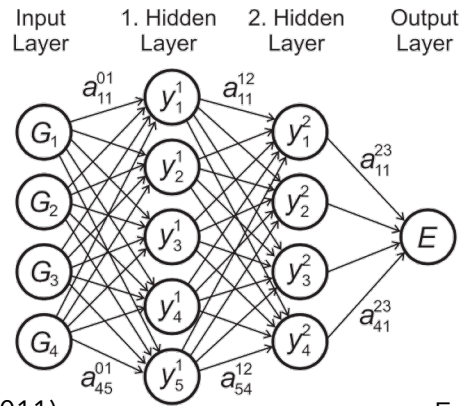
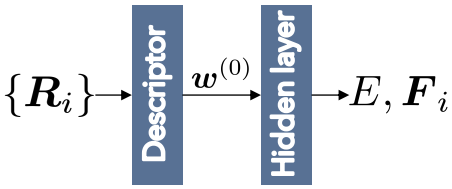


Figure from J. Chem. Phys. **134**, 074106 (2011)

Neuroevolution potential (NEP) models

Simple architecture



Optimization

separable natural evolution strategy

Loss function

$$\mathcal{L}(\boldsymbol{w}) = \lambda_{\text{energy}} \cdot \text{RMSE}(\text{energy}) + \lambda_{\text{forces}} \cdot \text{RMSE}(\text{forces}) + \lambda_{\text{virial}} \cdot \text{RMSE}(\text{virial}) + \lambda_1 \cdot \|\boldsymbol{w}\|_1 + \lambda_2 \cdot \|\boldsymbol{w}\|_2$$

The descriptor is part of the learning
e.g., for radial (pairwise) part

$$q_n^i = \sum_{j \neq i} g_n(r_{ij}) \quad \text{with} \quad 0 \le n \le n_{\max}^{\text{R}},$$

$$g_n(r_{ij}) = \sum_{k=0}^{N_{\text{bas}}^{\text{R}}} c_{nk}^{ij} f_k(r_{ij})$$

trained

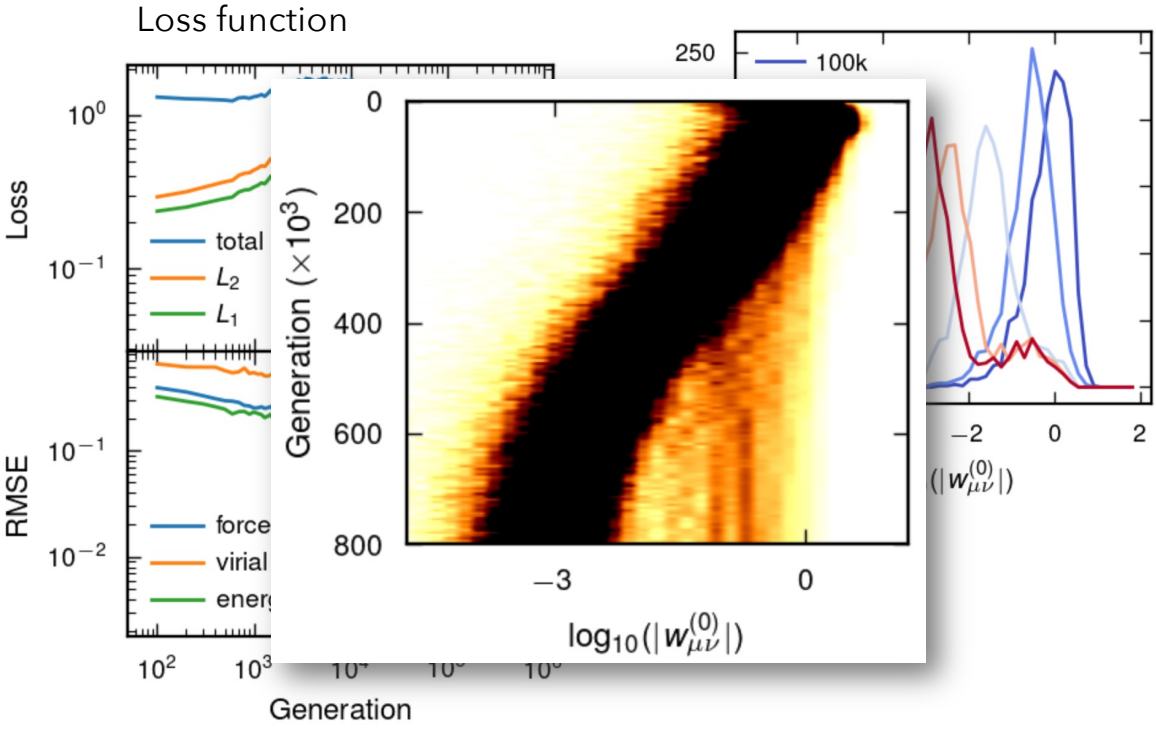
$$f_k(r_{ij}) = \frac{1}{2}\left[T_k\left(2\left(r_{ij}/r_{\text{c}}^{\text{R}} - 1\right)^2 - 1\right) + 1\right] f_{\text{c}}(r_{ij})$$

Regularization
→ stability
→ data efficiency

Fan *et al.*, J. Chem. Phys. 157, 114801 (2022)

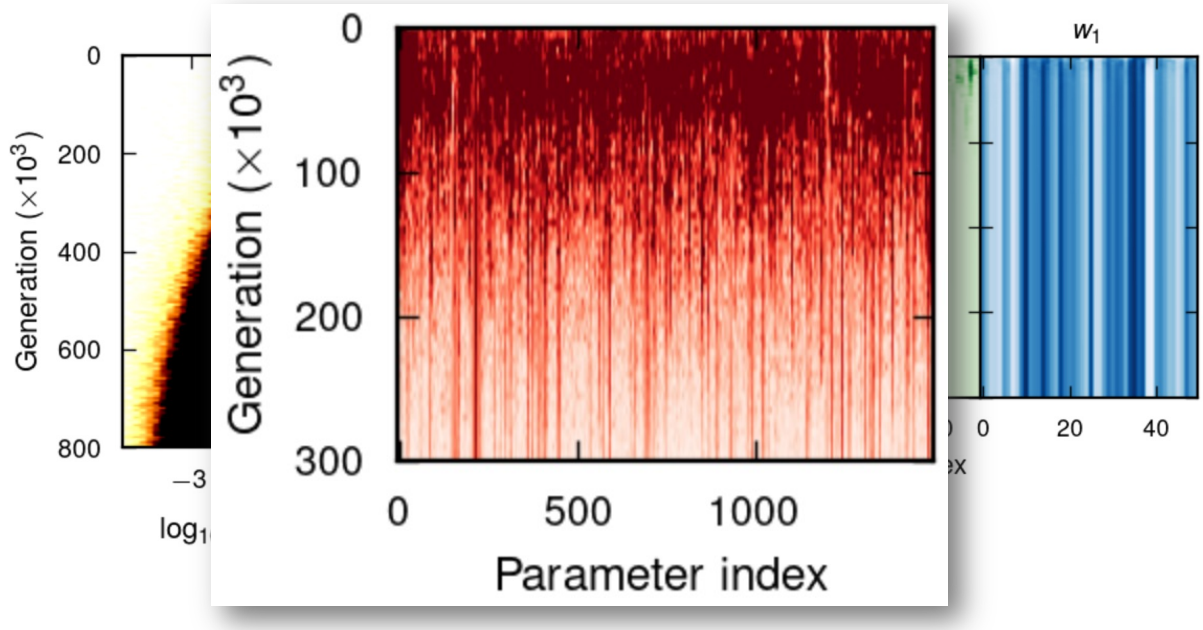
Evolution of sparsity during training

Loss function



Model for $Cs_{1-x}Rb_xPbCl_{3(1-y-z)}Br_{3y}I_{3z}$

Feature selection

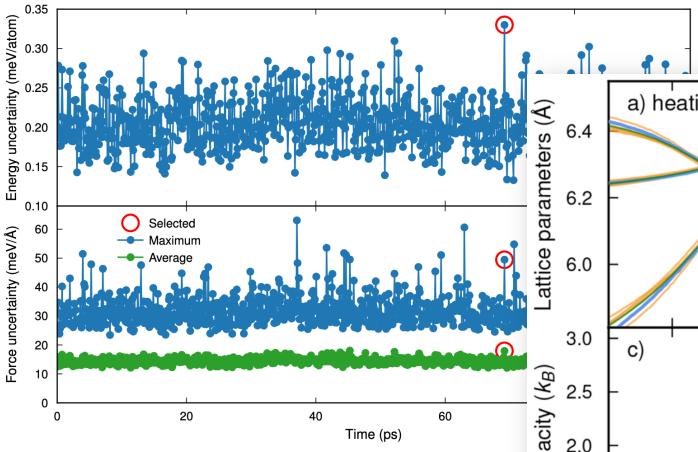


Note the "jumps" between solutions

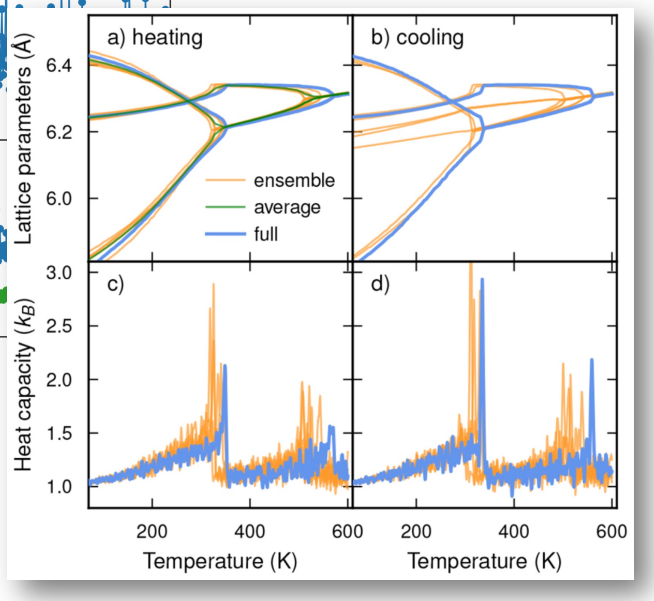
In the final model only 25% of the NN weights are non-zero

Model ensemble through k-fold

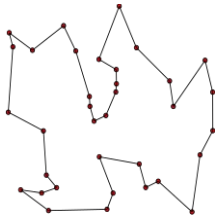
Active learning



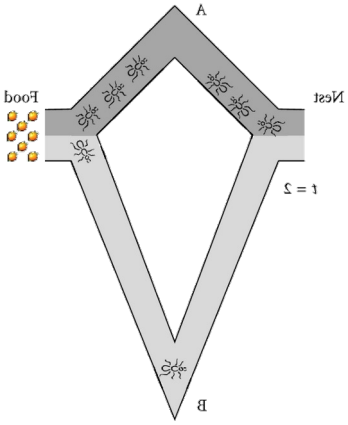
Ensemble predictions



Ant colony optimization

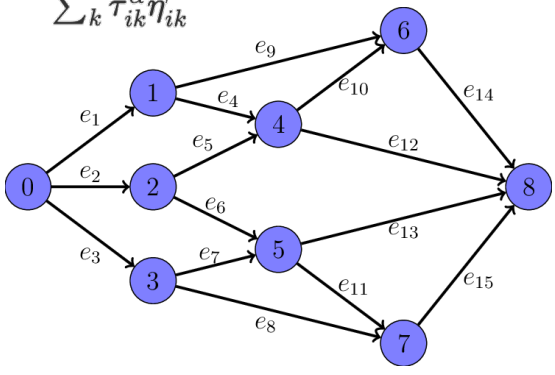


1. Walkers are placed at origin and| stochastically find pathways through graph
2. Quality (weighted length) of paths is calculated
3. “Pheromone” is deposited along each path, with more pheromone placed on shorter paths
4. Next set of walkers chooses each next step weighing in both the desirability (typically related to the length) of the path and the amount of pheromone already deposited



Ant colony optimization

Probability for choosing a step $\mathcal{P}_{ij} = \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_k \tau_{ik}^{\alpha}\eta_{ik}^{\beta}}$



Pheromone update

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \sum_k \Delta\tau_{ik}^k$$

$$\Delta\tau_{ik}^m = \begin{cases} \frac{Q}{L_m} & \text{if ant } m \text{ used edge } (i,k) \text{ during its current tour} \\ 0 & \text{otherwise} \end{cases}$$

Demo 1

Demo 2

α, β, Q are adjustable constants

Images from https://wikipedia.org

Response surface methodology-based

Replace complex objective with (very) cheap emulator
→ optimize using emulator via optimizer or simply grid search

Bayesian optimization: use priors over unknown functions
→ Gaussian processes or Krigging

Algorithm

1. find the next sampling point by optimizing the acquisition function
2. sample the objective function
3. add the sample to the data and update the GP
4. return to 1.

Choices

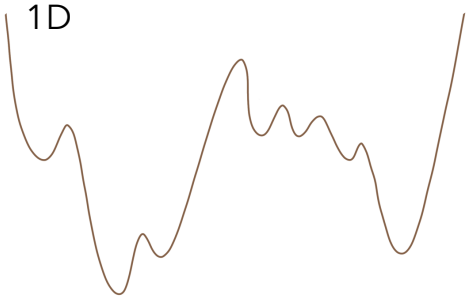
- dimensionality of space (descriptor, reaction coordinate, …)
- acquisition function/strategy

$$\boldsymbol{x}_{\text{new}} = \arg\min_{\boldsymbol{x}} \left[f(\boldsymbol{x}) + \beta \operatorname{var}_f(\boldsymbol{x}) \right]$$

β → 0: exploitation
β → ∞: exploration

Summary

1D



2D

Possible targets

Atomic structures
Model loss functions
Instrument settings
Path finding
Network optimization
…

Stochastic methods

Simulated annealing
Replica exchange

(Meta)heuristic methods

Evolutionary methods (including genetic algorithms)
Ant colony optimization
Particle swarm optimization

Response surface methodology-based

Bayesian optimization

Exploitation vs exploration