Sampling Can Be Faster Than Optimization

Central University of Finance and Economics

Jia Fan

Dec 14th , 2018

Outline:

- ☐ Introduction
- ☐ Polynomial Convergence of MCMC Algorithms
- ☐ Exponential Dependence on Dimension for Optimization
- ☐ Parameter Estimation from Gaussian Mixture Model
- ☐ Discussion

Introduction:

- ***Machine learning*** and ***data science*** are fields that blend computer science and statistics so as to solve inferential problems whose scale and complexity require modern computational infrastructure.
- The algorithmic foundations on which these blends have been based repose on two general computational strategies, both which have their roots in mathematics--***optimization*** and ***Markov chain Monte Carlo (MCMC) sampling***.

Introduction:

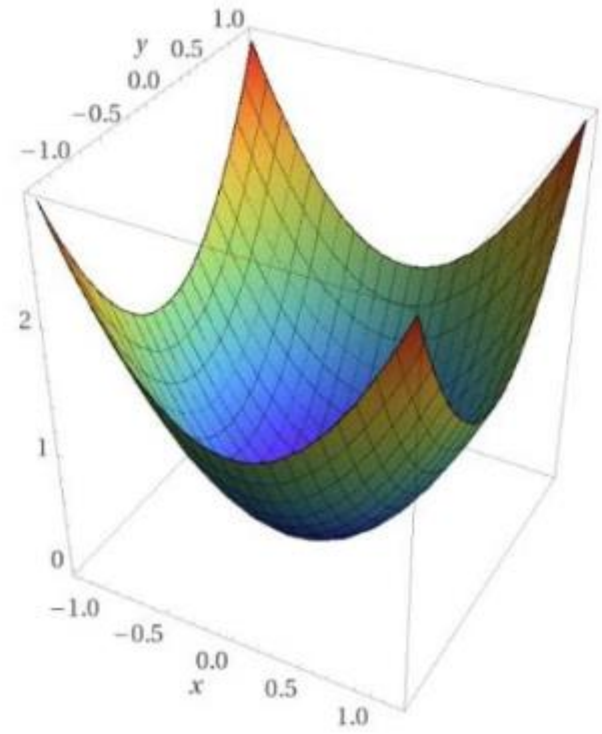
- Research on ***optimization*** focused on ***estimation and prediction*** problems. (SGD,EM)
- And research on ***sampling*** focused on tasks that require ***uncertainty estimates***, such as forming confidence intervals and conducting hypothesis tests.
- The relative paucity of theoretical research linking optimization and sampling has limited the flow of ideas.

Introduction:

- The overall message from the theoretical linkages have begun to appear in recent work is that ***sampling is slower than optimization***.
- Sampling approaches are warranted ***only if*** there is need for the ***stronger inferential outputs*** that they provide.

Introduction:

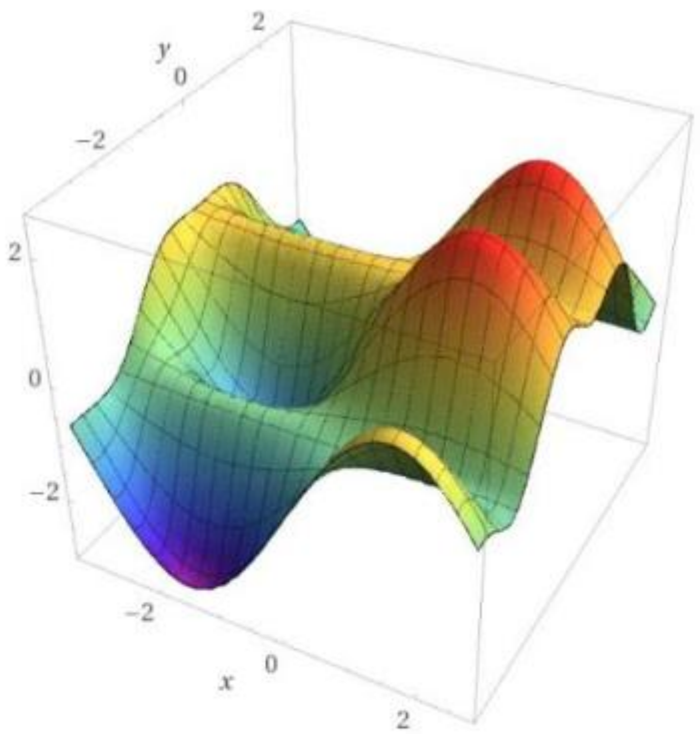
- These results are, however, obtained in the setting of ***convex functions***.
- For convex functions, global properties can be assessed via local information.
- Not surprisingly, gradient-based optimization is well suited to such a setting



Introduction:

- This paper considered

a broad class of problems that are ***strongly convex outside of a bounded region***, but ***nonconvex inside*** of it.



Polynomial Convergence of MCMC :

- **Assumptions on U :**

1. $U(\mathbf{x})$ is L-Lipschitz smooth and its Hessian exists $\forall \mathbf{x} \in \mathbb{R}^d$.

   That is: $U \in C^1(\mathbb{R}^d)$, $\forall \mathbf{x}, \mathbf{z} \in \mathbb{R}^d$, $\|\nabla U(\mathbf{x}) - \nabla U(\mathbf{z})\| \leq L \|\mathbf{x} - \mathbf{z}\|$; $\forall \mathbf{x} \in \mathbb{R}^d$, $\nabla^2 U(\mathbf{x})$ exists.

2. $U(\mathbf{x})$ is m-strongly convex for $\|\mathbf{x}\| > R$.

   That is: $V(\mathbf{x}) = U(\mathbf{x}) - \frac{m}{2} \|\mathbf{x}\|_2^2$ is convex on $\Omega = \mathbb{R}^d \setminus \mathbb{B}(0, R)$ [1]. We then follow the definition of convexity on nonconvex domains [38, 46] to require that $\forall \mathbf{x} \in \Omega$, any convex combination of $\mathbf{x} = \lambda_1 \mathbf{x}_1 + \cdots + \lambda_k \mathbf{x}_k$ with $\mathbf{x}_1, \cdots, \mathbf{x}_k \in \Omega$ satisfy:

   $$V(\mathbf{x}) \leq \lambda_1 V(\mathbf{x}_1) + \cdots + \lambda_k V(\mathbf{x}_k).$$

   We further denote the "condition number" of U on Ω as $\kappa = L/m$.

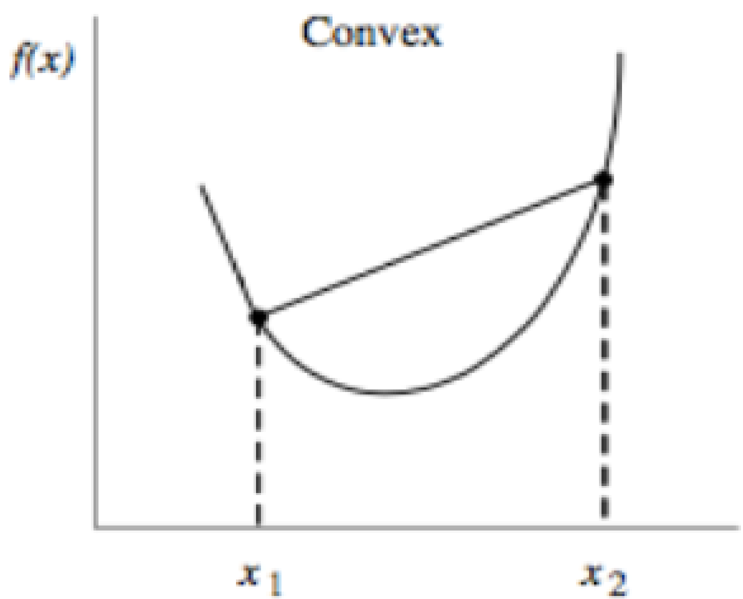
3. For convenience, let $\nabla U(0) = 0$ (i.e., 0 is a local extremum).

Polynomial Convergence of MCMC :

- **convex**

A function $f(x): A \to \mathbb{R}$ is convex if :

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y), \lambda \in [0,1]$$



Polynomial Convergence of MCMC :

- **Langevin Algorithm (ULA)**

  -- a family of gradient-based MCMC sampling algorithms

- **Pseudocode**

Input: $\mathbf{x}^0$, stepsizes $\{h^k\}$
for $k = 0, 1, 2, \ldots, K-1$ **do**
$\quad \mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - h^k \nabla U(\mathbf{x}^k) + \xi$
$\quad$ **if** $\dfrac{p\left(\mathbf{x}^k | \mathbf{x}^{k+1}\right) p^*\left(\mathbf{x}^k\right)}{p\left(\mathbf{x}^{k+1} | \mathbf{x}^k\right) p^*\left(\mathbf{x}^{k+1}\right)} < u$ **then**
$\quad\quad \mathbf{x}^{k+1} \leftarrow \mathbf{x}^k$
Return $\mathbf{x}^K$

Metropolis Adjusted Langevin Algorithm (MALA)

Polynomial Convergence of MCMC :

- **Langevin Algorithm (ULA)**

Input: $\mathbf{x}^0$, stepsizes $\{h^k\}$
for $k = 0, 1, 2, \ldots, K-1$ **do**
$\quad \mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - h^k \nabla U(\mathbf{x}^k) + \xi$
$\quad$ **if** $\dfrac{p\left(\mathbf{x}^k|\mathbf{x}^{k+1}\right) p^*(\mathbf{x}^k)}{p\left(\mathbf{x}^{k+1}|\mathbf{x}^k\right) p^*\left(\mathbf{x}^{k+1}\right)} < u$ **then**
$\quad\quad \mathbf{x}^{k+1} \leftarrow \mathbf{x}^k$
Return $\mathbf{x}^K$

- **Gradient Descent**

Input: $\mathbf{x}^0$, stepsizes $\{h^k\}$
for $k = 0, 1, 2, \ldots, K-1$ **do**
$\quad \mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - h^k \nabla U(\mathbf{x}^k)$
Return $\mathbf{x}^K$

- It is essentially the same as gradient descent, differing only in its incorporation of a random term in the update $\xi \sim \mathcal{N}(0, 2h^k \mathbb{I})$

Polynomial Convergence of MCMC :

- Sampling from a smooth target distribution p^* that is strongly log-concave outside of a region

- Define the ε-mixing time in total variation distance as

$$\tau(\epsilon; p^0) = \min\left\{k \middle| \left\|p^k - p^*\right\|_{\mathrm{TV}} \leq \epsilon\right\}.$$

Polynomial Convergence of MCMC :

- **demonstration**

- First, use properties of $p^* \propto e^{-U}$ to establish linear convergence of a continuous stochastic process that underlies Algorithm 1.

- Then discretize, finding an appropriate step size for the algorithm to converge to the desired accuracy.

Polynomial Convergence of MCMC :

• Kullback-Leibler Divergence

• KLD upper bounds the total variation distance and allows us to obtain strong convergence guarantees that include dimension dependence.

$$D_{\mathrm{KL}}(P \parallel Q) = -\sum_i P(i) \log\left(\frac{Q(i)}{P(i)}\right),$$

$$D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx,$$

Polynomial Convergence of MCMC :

Theorem 1. *For $p^* \propto e^{-U}$, we assume that U is m-strongly convex outside of a region of radius R and L-Lipschitz smooth (see the Supplement for a formal statement of the assumptions). Let $\kappa = L/m$ denote the condition number of U. Consider Algorithm $\boxed{1}$ with initialization $p^0 = \mathcal{N}\left(0, \frac{1}{L}\mathbb{I}_d\right)$ and error tolerance $\epsilon \in (0,1)$. Then ULA satisfies*

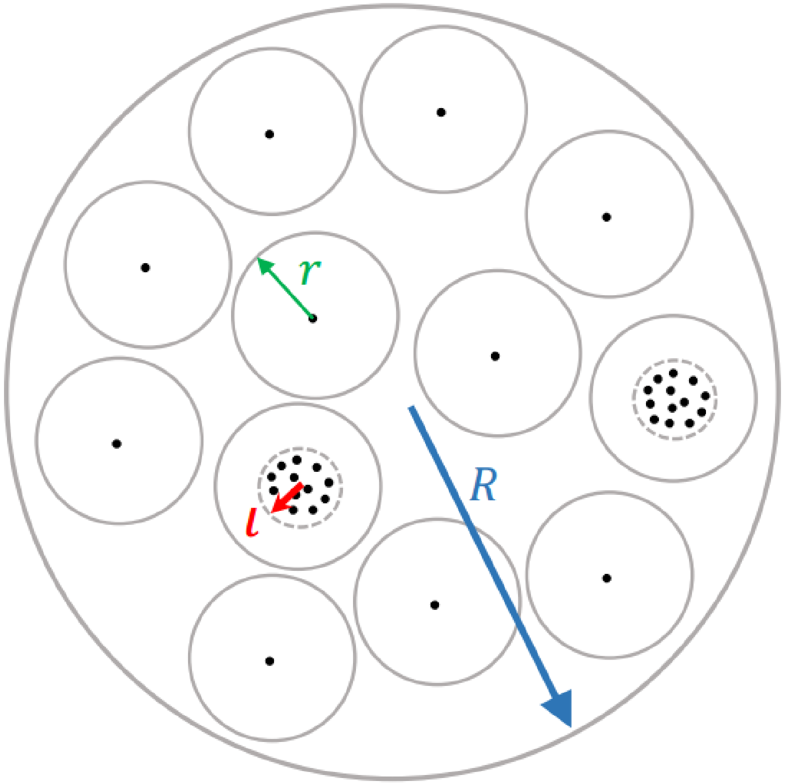
$$\tau_{ULA}(\epsilon, p^0) \leq \mathcal{O}\left(e^{32LR^2}\kappa^2 \frac{d}{\epsilon^2} \ln\left(\frac{d}{\epsilon^2}\right)\right). \tag{1}$$

For MALA,

$$\tau_{MALA}(\epsilon, p^0) \leq \mathcal{O}\left(e^{16LR^2}\kappa^{3/2} d^{1/2}\left(d \ln \kappa + \ln\left(\frac{1}{\epsilon}\right)\right)^{3/2}\right). \tag{2}$$

Exponential Dependence on Dimension for Optimization

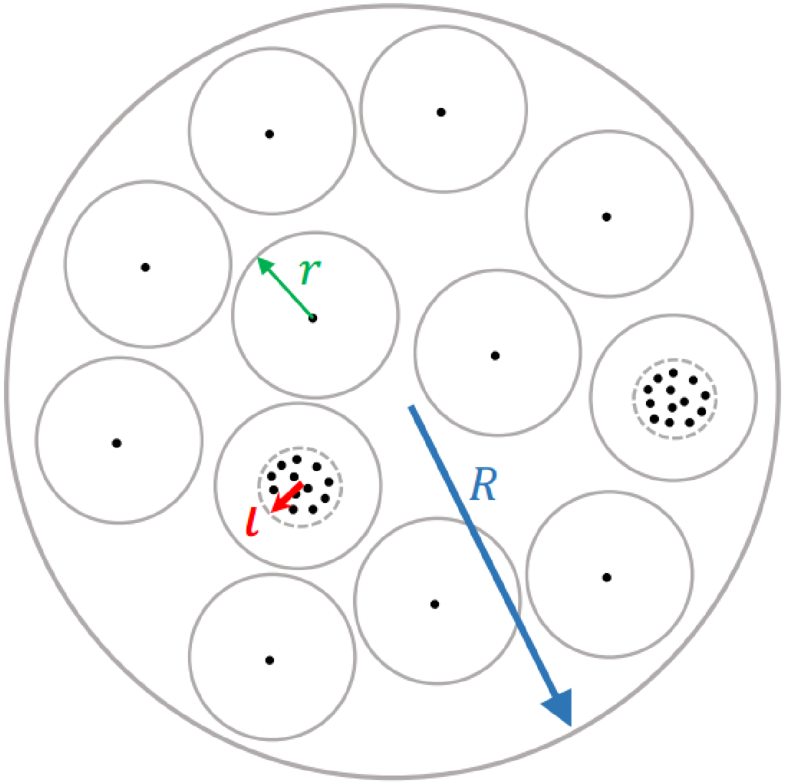
- Consider a general iterative algorithm family A which, at every step k, is allowed to query not only the function value of U but also its derivatives up to any fixed order at a chosen point x^k. Thus the algorithm has access to the vector($\{U(x^k), \nabla U(x^k), \ldots, \nabla^n U(x^k)\}$,for any fixed $n \in \mathcal{N}$)



Exponential Dependence on Dimension for Optimization

Theorem 2 (Lower bound for optimization). *For any $R > 0$, $L \geq 2m > 0$, and $\epsilon \leq \mathcal{O}(LR^2)$, there exists an objective function, $U : \mathbb{R}^d \to \mathbb{R}$, which is m-strongly convex outside of a region of radius $2R$ and L-Lipschitz smooth, such that any algorithm in $\mathcal{A}$ requires at least $K = \Omega((LR^2/\epsilon)^{d/2})$ iterations to guarantee that $\min_{k \leq K} |U(\mathbf{x}^K) - U(\mathbf{x}^*)| < \epsilon$ with constant probability.*

- A depiction of an example
- Randomly assign the minimum $x^*$to one of the balls, assigning a larger constant value to the other balls.
- The number of queries needed to find the specific ball containing the minimum is exponential in d.



Parameter Estimation from Gaussian Mixture Model:

Sampling

- Inferring the mean parameters of a Gaussian mixture model $\mu = \{\mu_1, \dots, \mu_M\} \in R^{d\times M}$

$$p\left(y_n|\boldsymbol{\mu}\right) = \sum_{i=1}^{M} \frac{\lambda_i}{Z_i} \exp\left(-\frac{1}{2}(y_n - \mu_i)^{\mathrm{T}}\Sigma_i^{-1}(y_n - \mu_i)\right) + \left(1 - \sum_{i=1}^{M} \lambda_i\right) p_0(y_n),$$

- where Z_i are normalization constants and $\sum_{i=1}^{M} \lambda_i \leq 1$

- $p_o(y_0)$ represents general constraints on the data

Parameter Estimation from Gaussian Mixture Model:

Sampling

- The objective function is given by the log posterior distribution:

$$U(\boldsymbol{\mu}) = -\log p(\boldsymbol{\mu}) - \sum_{n=1}^{N} \log p\left(y_n|\boldsymbol{\mu}\right)$$

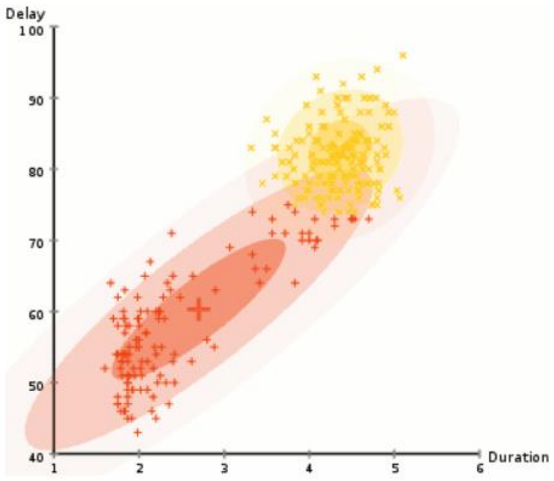
- For a suitable choice of the prior p_μ and weights $\{\lambda_i\}$, the objective function is Lipschitz smooth and strongly convex for $||\mu|| \geq R\sqrt{M}$.

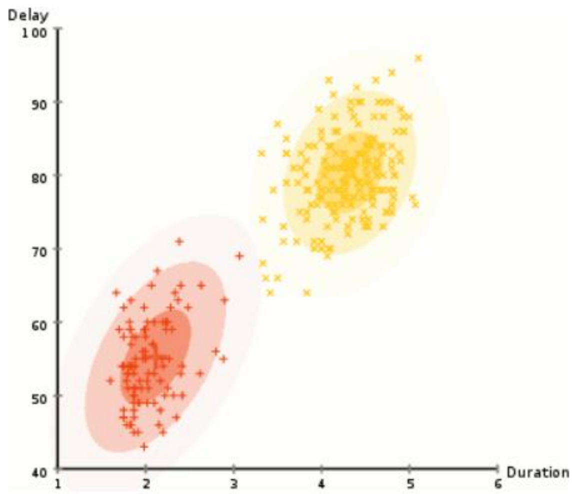
- Therefore, taking $MR^2 = \mathcal{O}(\log d)$, the ULA and MALA algorithms converge to ϵ accuracy within $\mathrm{K} \leq \tilde{\mathcal{O}}\ (d^3/\epsilon)$ and $\mathrm{K} \leq \tilde{\mathcal{O}}\ (d^3 ln^2(\frac{1}{\epsilon}))$ steps, respectively.

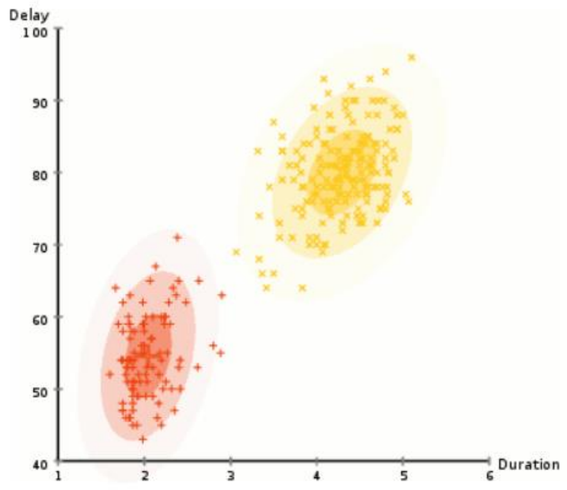
Parameter Estimation from Gaussian Mixture Model:

Optimization

- **EM algorithm (Expectation-Maximum)**







Parameter Estimation from Gaussian Mixture Model:

Optimization

- **EM algorithm (Expectation-Maximum)**

- 1. Initializes the distribution parameters
- 2. Repeat until convergence:

(E-step) For each i, set

$$Q_i(z^{(i)}) := p(z^{(i)}|x^{(i)};\theta).$$

(M-step) Set

$$\theta := \arg\max_{\theta} \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})}.$$

Parameter Estimation from Gaussian Mixture Model:

Optimization

- **EM algorithm (Expectation-Maximum)**
- Initialize the EM algorithm by randomly selecting M data points (sometimes with small perturbations) to form μ_0.
- (E) step a weight is computed for each data point and each mixture component, using the current parameter value μ_{k}.
- (M) step the value of $\mu_{\mathrm{k}+1}$ is updated as a weighted sample mean

Parameter Estimation from Gaussian Mixture Model:

Optimization

- **EM algorithm (Expectation-Maximum)**

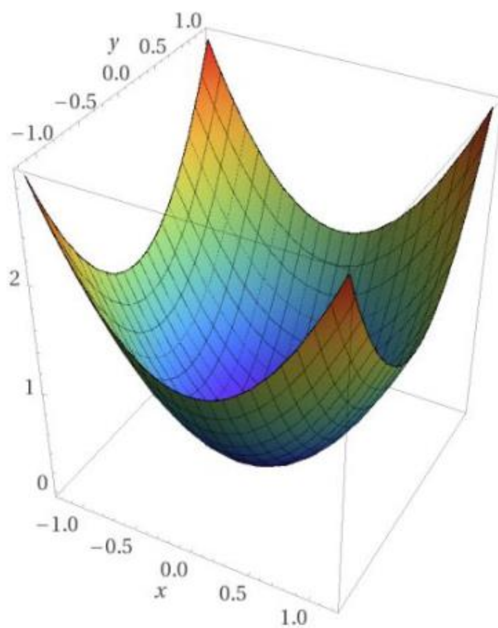
- Under the condition that $MR^2 = \mathcal{O}(\log d)$, the EM algorithm requires more than$\mathrm{K} \geq \min\{\mathcal{O}\left(d^{1/\epsilon}\right), \mathcal{O}\left(d^{d}\right)\}$, queries to converge if one initializes the algorithm close to the given data points.

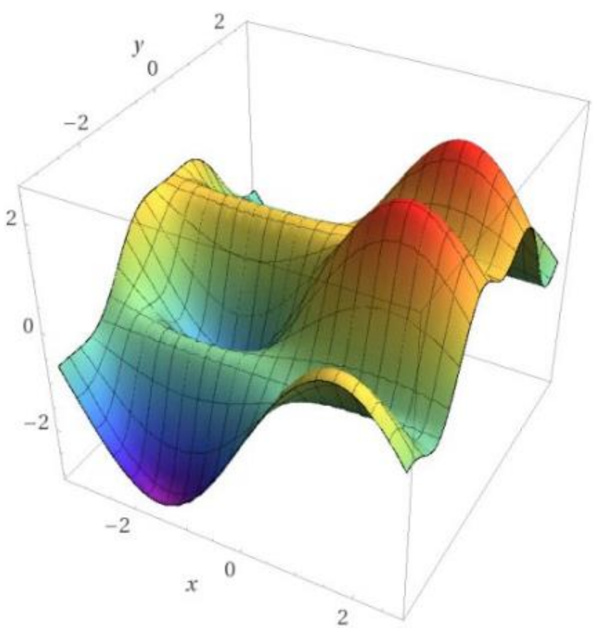
Discussion

- A natural family of ***nonconvex*** functions for which ***sampling algorithms*** have polynomial complexity in dimension whereas ***optimization algorithms*** display exponential complexity.
- The intuition behind these results is that computational complexity for optimization algorithms depends heavily on the local properties of the objective function U.

Discussion

- ***Sampling*** complexity depends more heavily on the ***global*** properties of *U*.
- ***Optimization*** algorithms complexity depends heavily on the ***local*** properties of the objective function U--local strong convexity near the global optimum can improve the convergence rate of convex optimization.





What we have learnt:

- **The conclusion—Sampling or optimization?**
- **The setting of a research object (Nonconvex)**
- **The process of deducing the argument (Fig.)**
- **Writing procedures and rules(the main body and the appendix)**