Reinforcement Learning China Summer School

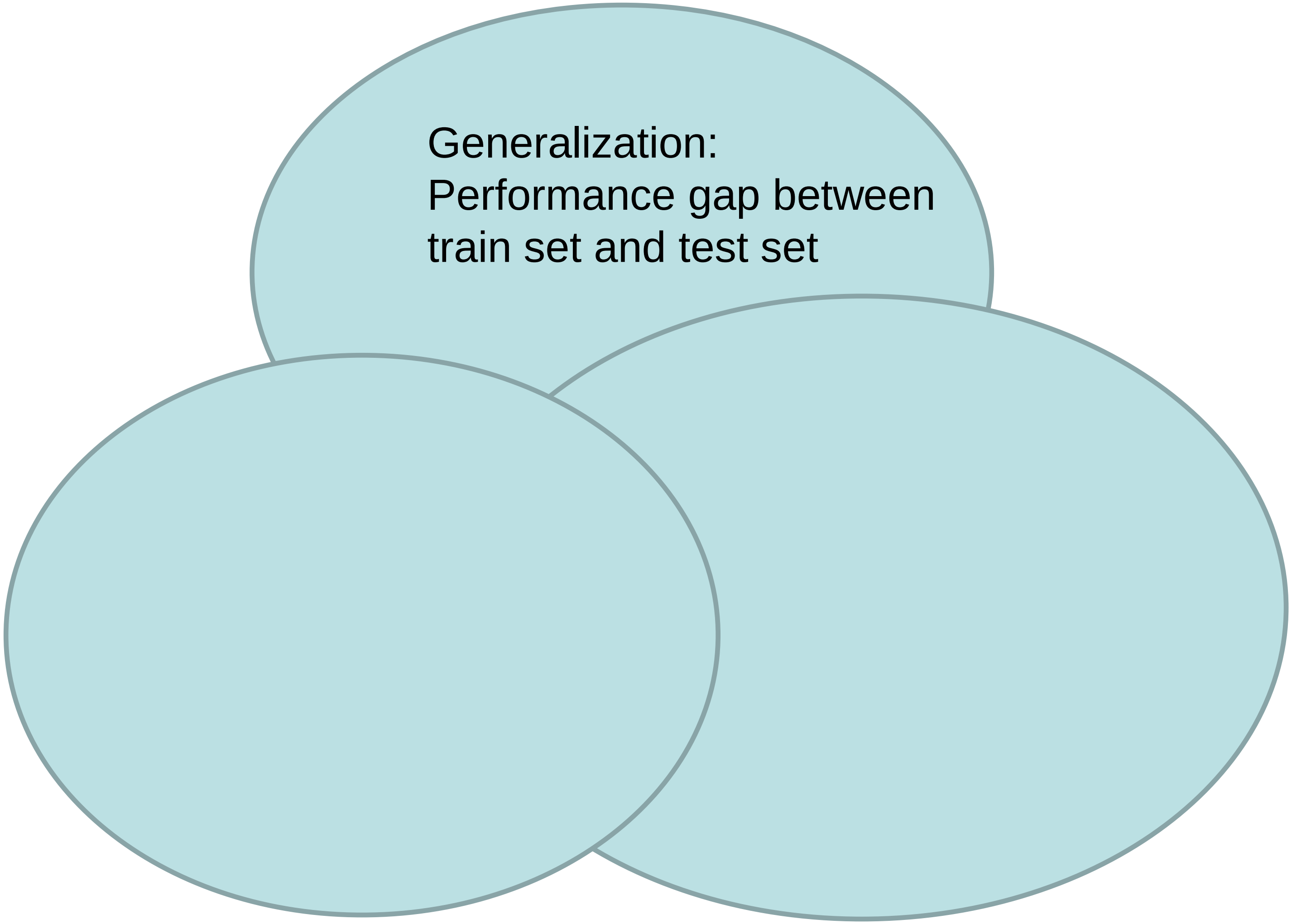
RLChina 2021

A Brief Introduction to Optimization for Machine Learning

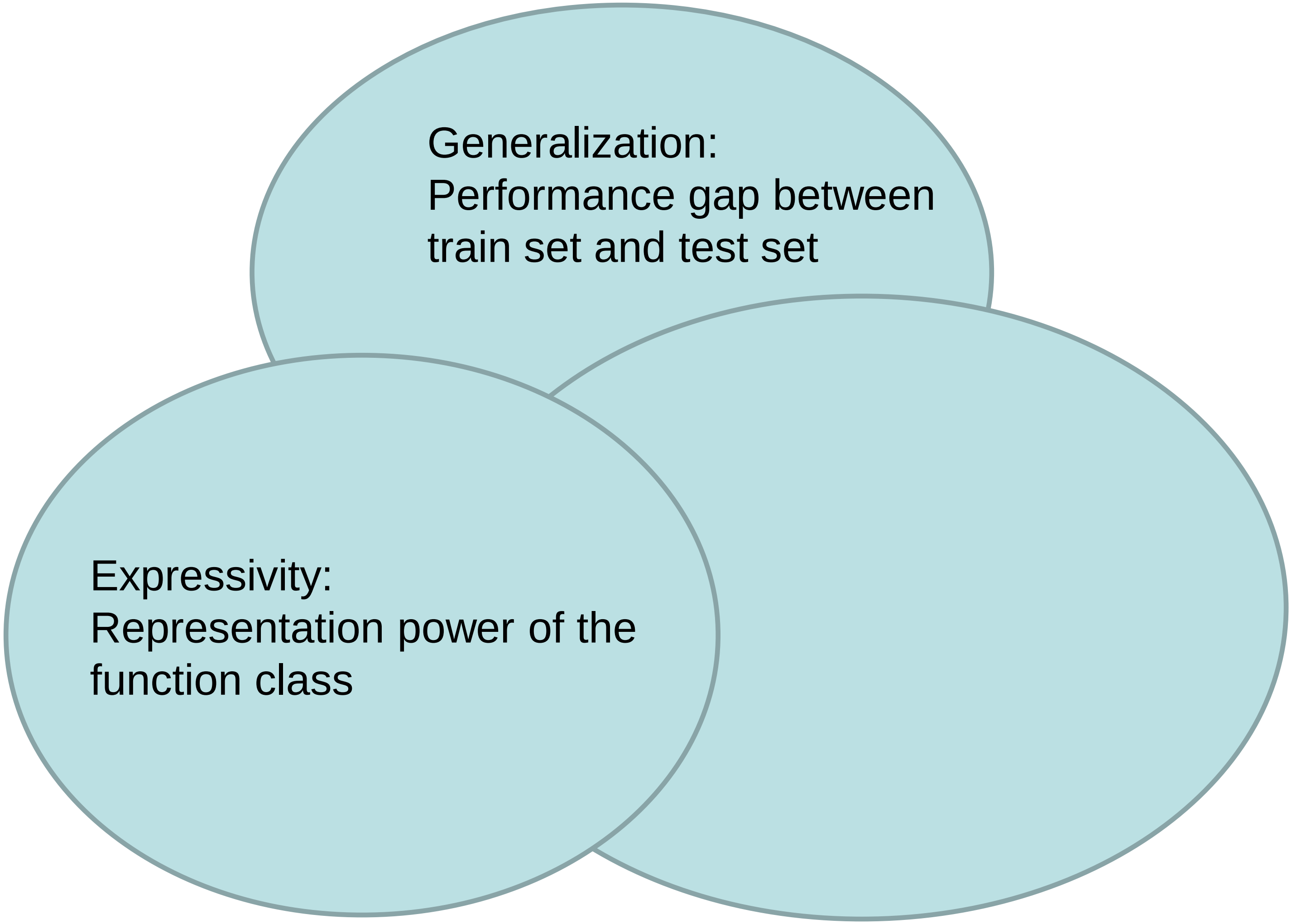
Jingzhao Zhang

August 16, 2021

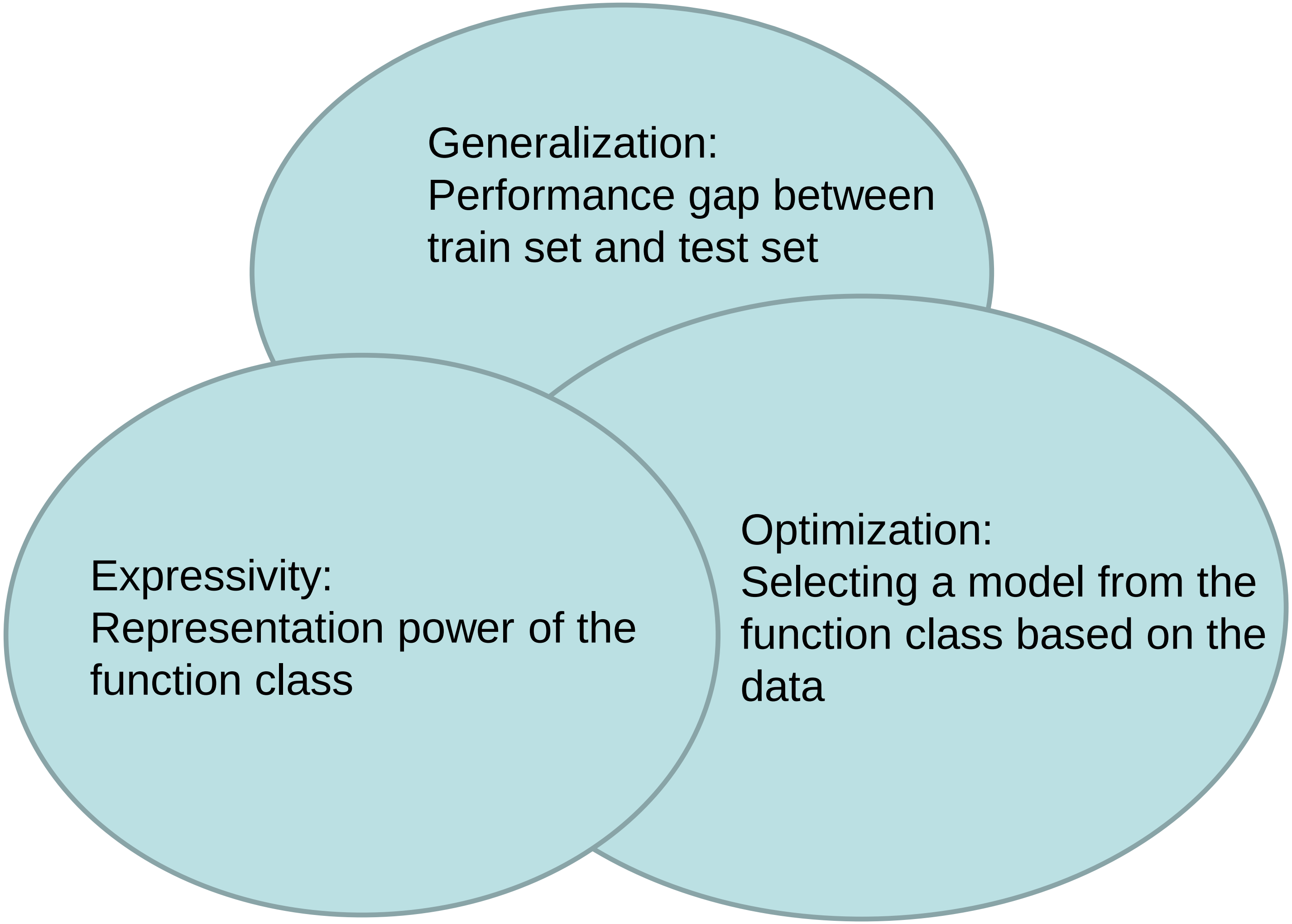
Machine Learning Theory



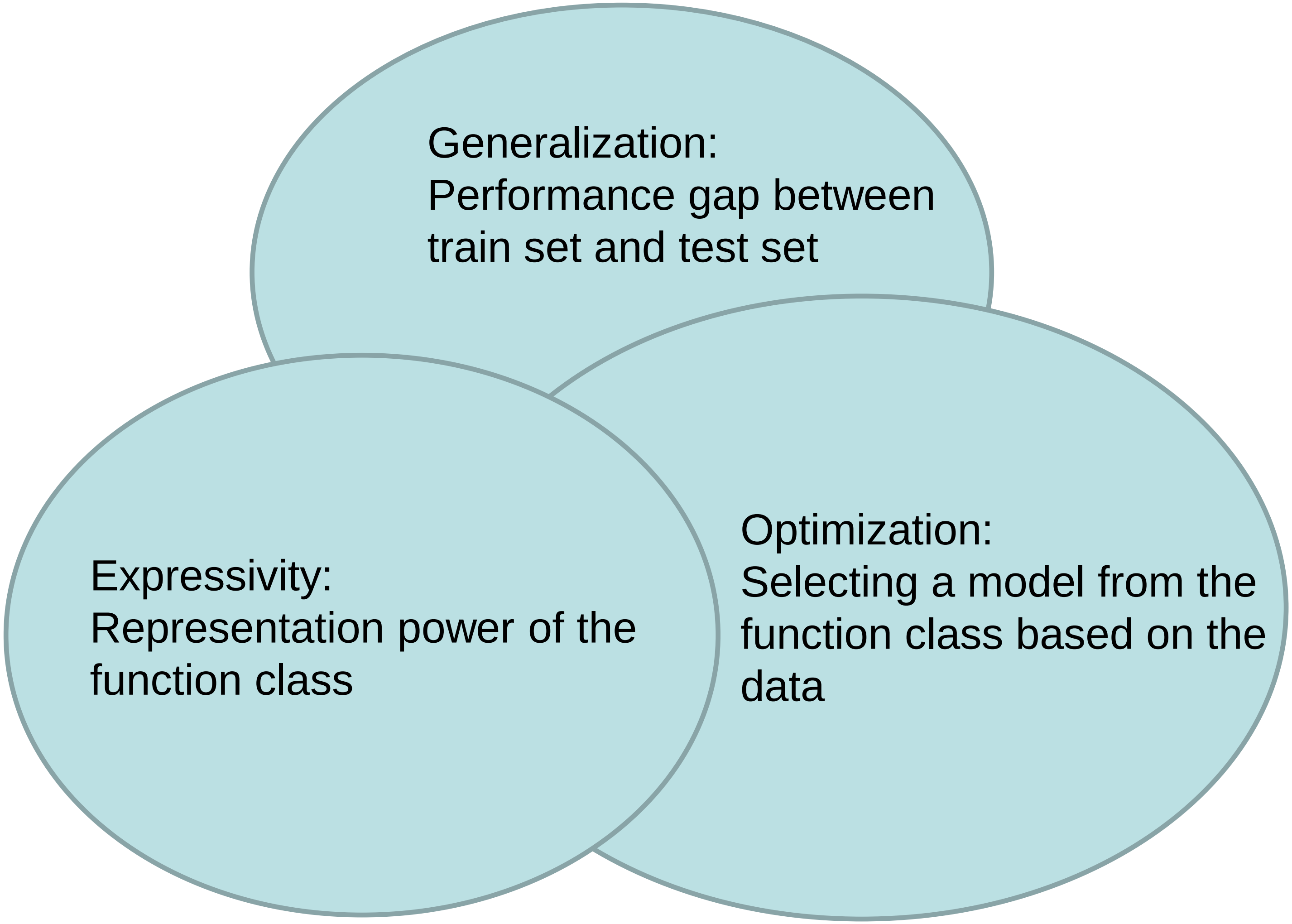
Machine Learning Theory



Machine Learning Theory

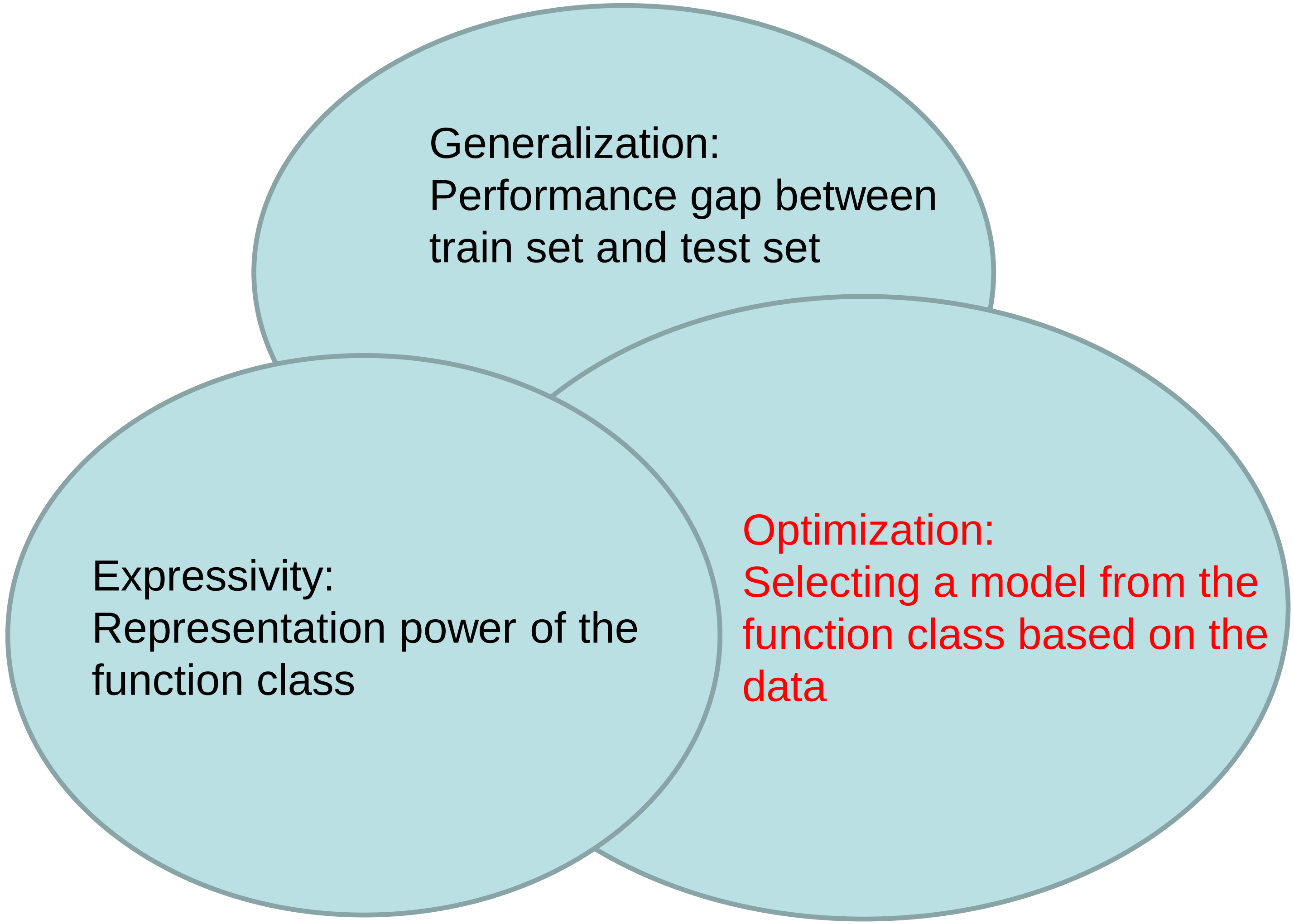


Machine Learning Theory



Active research topics are at the intersections.

Machine Learning Theory



Outline

- Optimization Algorithms
- Convergence analysis: Gradient Methods
- Graphical Model & Bayesian Inference
- Bayesian Optimization

Continuous Optimization problem

- Formulation

$$\min_{x \in \mathcal{X}} \; f(x)$$

Continuous Optimization problem

- Formulation

$$\min_{x \in \mathcal{X}} \; f(x)$$

Variable

Constraint

Objective

Continuous Optimization problem

- Formulation

$$\min_{x \in \mathcal{X}} \; f(x)$$

- Examples: Neural network training

$$\underset{x \in \mathbb{R}^p}{\text{minimize}}\, f(x) := \frac{1}{n} \sum_{i=1}^{n} l_i(x) + \lambda \Omega(x)$$

Neural network params

Samples

Regularizers

Continuous Optimization problem

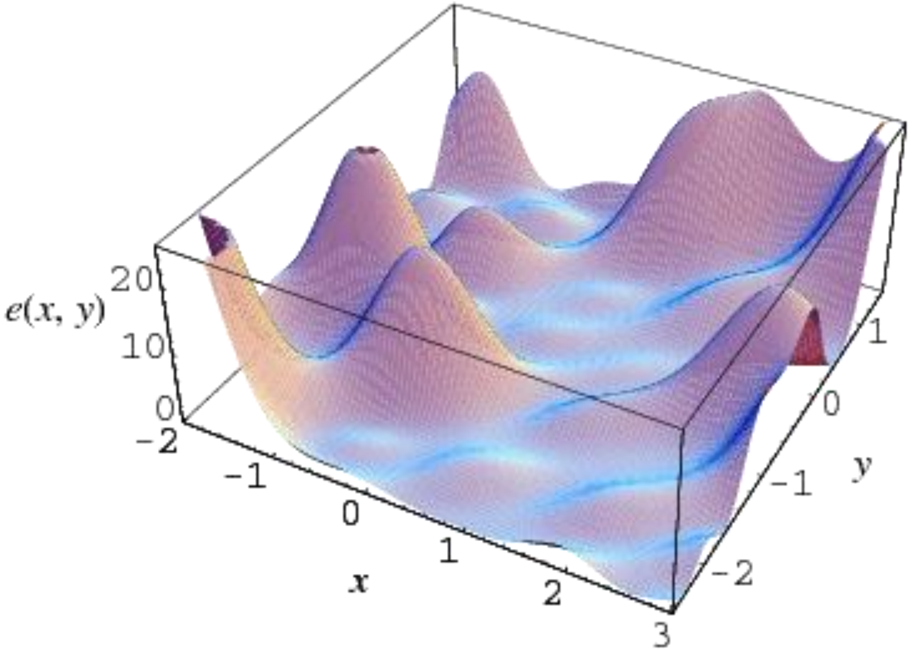
$$\min_{x \in \mathcal{X}} \; f(x)$$

Optimization Algorithm

Solution: x*

Optimization Algorithm: 0th order

- Gridding



https://mathworld.wolfram.com/GlobalOptimization.html

Optimization Algorithm: 0^{th} order

- Sampling (Metropolis Hastings, evolutionary algorithm,

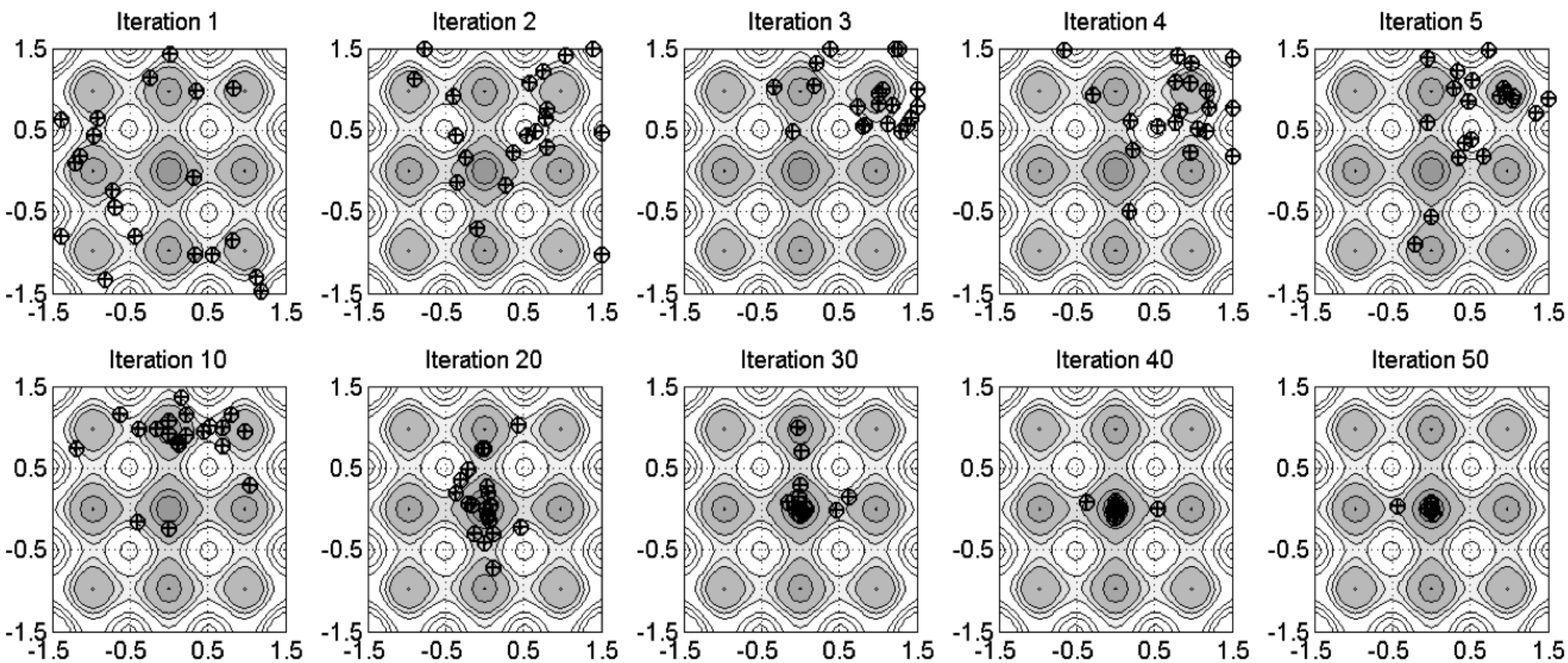
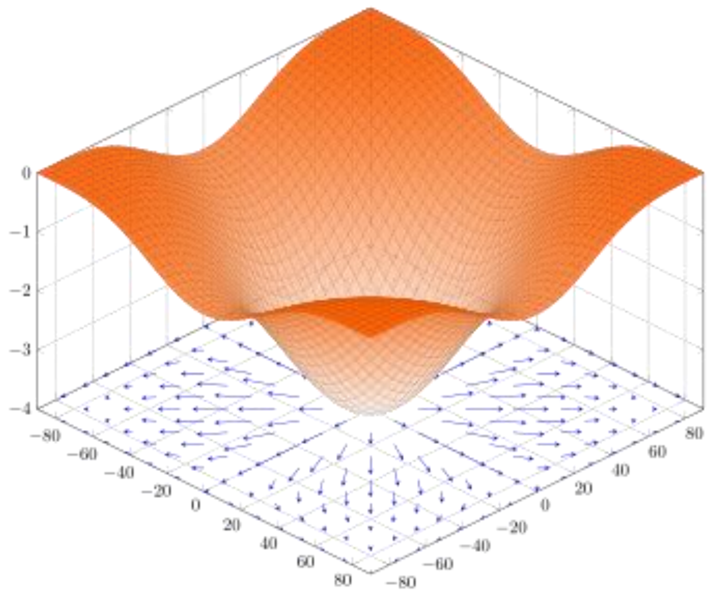


Figure 6. Example of particle positions with respect to the iteration.

https://www.sft.asso.fr/Local/sft/dir/user-3775/documents/actes/journeessft/metti_5_2012/Lectures&Tutorials-Texts/Text-T2-Ruffio.pdf

Optimization Algorithm: 1st Order

- 0th order queries function value only is easy to implement but slow.



- 1st order method uses gradient information.
- 1st method is much faster and works well with backprop.

Optimization Algorithm: 1^{st} Order

- 0th order is easy to implement but slow.
- 1^{st} method is much faster and works well with backprop.

- SGD: Vanilla

$$\theta_{t+1} = \theta_t - \eta g_t$$

- Adagrad: Coordinate-wise

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t.$$

- ADAM: Momentum + Coordinate-wise

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t.$$

Optimization Algorithm: Higher Order

- Newton's method

$$x_{t+1} = x_t - \eta \nabla^2 f(x_t)^{-1} \nabla f(x_t)$$

Optimization Algorithm: Higher Order

- Newton's method

$$x_{t+1} = x_t - \eta \nabla^2 f(x_t)^{-1} \nabla f(x_t)$$

- Requires expensive computations.
- Faster convergence.
- Works well in low dimensional problems.

Outline

- Optimization Algorithms
- Convergence analysis: Gradient Methods
- Graphical Model & Bayesian Inference
- Bayesian Optimization

Convergence Analysis: Oracle Complexity

A simple analysis of stochastic gradient descent for nonconvex functions.

$$x_{k+1} = x_k - \eta g_k$$

Algorithm definition

Oracle: gradient (1st order)

Convergence Analysis: Oracle Complexity

A simple analysis of stochastic gradient descent for nonconvex functions.

$$x_{k+1} = x_k - \eta g_k$$

Algorithm definition

Oracle: gradient (1st order)

We want to prove how fast the process can find to a stationary point.

$$\|\nabla f(x_k)\| \leq \epsilon$$

Optimality measure

A key lemma to prove convergence

Definition 8.1 (L-smooth) *A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be L-smooth is for all $x, y \in \mathbb{R}^n$, we have that*

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L \|x - y\|$$

A key lemma to prove convergence

Definition 8.1 (L-smooth) *A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be L-smooth is for all $x, y \in \mathbb{R}^n$, we have that*

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L \|x - y\|$$

Lemma 8.2 *If $f : \mathbb{R}^n \to \mathbb{R}$ be L-smooth. Then for all $x, y \in \mathbb{R}^n$ we have that*

$$|f(y) - (f(x) + \nabla f(x)^T (y - x))| \leq \frac{L}{2} \|x - y\|_2^2$$

A key lemma to prove convergence

Definition 8.1 (L-smooth) *A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be L-smooth is for all $x, y \in \mathbb{R}^n$, we have that*

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L \|x - y\|$$

Lemma 8.2 *If $f : \mathbb{R}^n \to \mathbb{R}$ be L-smooth. Then for all $x, y \in \mathbb{R}^n$ we have that*

$$|f(y) - (f(x) + \nabla f(x)^T (y - x))| \leq \frac{L}{2} \|x - y\|_2^2$$

Proof: Taylor expansion.

$$f(x) - f(y) - \nabla f(x)^T (y - x) = \int_0^1 (\nabla f(x_t) - \nabla f(x))^T (y - x) dt$$

See [Nesterov, Lectures on convex optimization]

Convergence Analysis: Oracle Complexity

A simple analysis of stochastic gradient descent

$$x_{k+1} = x_k - \eta g_k$$

$$\mathbb{E}[f(x_{k+1})] \le f(x_k) - \mathbb{E}[\langle \nabla f(x_k), x_{k+1} - x_k \rangle] + \frac{L\eta^2}{2}\mathbb{E}[\|g_k\|^2]$$

Function class

Differentiability L-smoothness

Convergence Analysis: Oracle Complexity

A simple analysis of stochastic gradient descent

$$x_{k+1} = x_k - \eta g_k$$

$$\mathbb{E}[f(x_{k+1})] \leq f(x_k) - \mathbb{E}[\langle \nabla f(x_k), x_{k+1} - x_k \rangle] + \frac{L\eta^2}{2}\mathbb{E}[\|g_k\|^2]$$

Function class

$$\mathbb{E}[f(x_{k+1})] \leq f(x_k) - \eta \|\nabla f(x_k)\| + \frac{\eta^2 L G^2}{2}$$

Oracle call properties

Unbiased, Bounded second moment

Convergence Analysis: Oracle Complexity

A simple analysis of stochastic gradient descent

$$x_{k+1} = x_k - \eta g_k$$

$$\mathbb{E}[f(x_{k+1})] \leq f(x_k) - \mathbb{E}[\langle \nabla f(x_k), x_{k+1} - x_k \rangle] + \frac{L\eta^2}{2}\mathbb{E}[\|g_k\|^2]$$

Function class

Oracle call properties

Unbiased, Bounded second moment

$$\mathbb{E}[f(x_{k+1})] \leq f(x_k) - \eta\|\nabla f(x_k)\| + \frac{\eta^2 LG^2}{2}$$

$$\eta \propto \frac{1}{\sqrt{T}} \implies \min_k \|\nabla f(x_k)\|^2 \leq \mathcal{O}(\frac{1}{\sqrt{T}})$$

Convergence Analysis: Oracle Complexity

A simple analysis of stochastic gradient descent

$$x_{k+1} = x_k - \eta g_k$$

$$\mathbb{E}[f(x_{k+1})] \le f(x_k) - \mathbb{E}[\langle \nabla f(x_k), x_{k+1} - x_k \rangle] + \frac{L\eta^2}{2}\mathbb{E}[\|g_k\|^2]$$

Function class

Oracle call properties

Unbiased, Bounded second moment

$$\mathbb{E}[f(x_{k+1})] \le f(x_k) - \eta\|\nabla f(x_k)\| + \frac{\eta^2 LG^2}{2}$$

$$\eta \propto \frac{1}{\sqrt{T}} \implies \min_k \|\nabla f(x_k)\|^2 \le \mathcal{O}(\frac{1}{\sqrt{T}})$$

Worst case performance (min-max rate)

$$\mathcal{T}_\epsilon(\mathcal{A}, \mathcal{F}) := \inf_{\mathsf{A} \in \mathcal{A}} \sup_{f \in \mathcal{F}} \mathsf{T}_\epsilon(\mathsf{A}, f).$$

Carmon, Yair, et al. "Lower bounds for finding stationary points i." *arXiv preprint arXiv:1710.11606* (2017).

Oracle Complexity: Key components

Oracle definition

Oracle properties

Function class

Optimality measure

Worst case performance

Optimality of SGD

1. The short analysis we did for SGD was theoretically optimal.

Optimality of SGD

1. The short analysis we did for SGD was theoretically optimal.

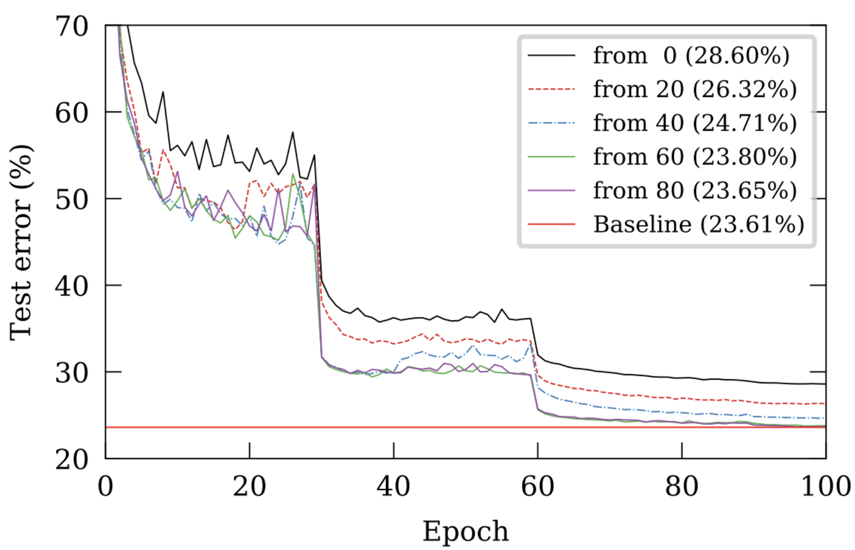
2. Practical algorithms such as SGD with momentum is theoretically slow

$$x_{k+1} = x_k + \beta(x_k - x_{k-1}) - \eta g_k$$

$$\min_{k=0,\ldots,t} \mathrm{E}[\|\nabla f(\mathbf{x}_k)\|^2] \leq \frac{2(f(\mathbf{x}_0) - f_*)(1-\beta)}{t+1} \max\left\{\frac{2L}{1-\beta}, \frac{\sqrt{t+1}}{C}\right\} + \frac{C}{\sqrt{t+1}} \frac{L\beta^2((1-\beta)s-1)^2(G^2+\sigma^2) + L\sigma^2(1-\beta)^2}{(1-\beta)^3}$$

Variance reduction algorithms

- For empirical risk minimization, theoretically faster algorithms (e.g. SAG, SAGA, SVRG, Spider) are designed.
- But they actually hurts practical performance…

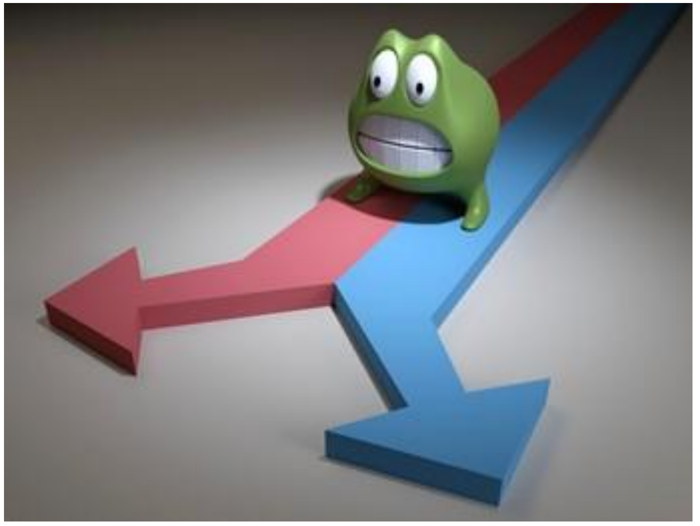


(a) ResNet-50

Defazio A, Bottou L. On the ineffectiveness of variance reduced optimization for deep learning[J]. arXiv preprint arXiv:1812.04529, 2018.

Closing the gap between theory and practice is still an active research area.

Theoretically fast algorithms



Empirically fast algorithms

On Complexity of Finding Stationary Points of Nonsmooth Nonconvex Functions, ICML 2020
Why ADAM Beats SGD for Attention Models, Neurips 2020
Why gradient clipping accelerates training: A theoretical justification for adaptivity, ICLR 2020

Outline

- Optimization Algorithms
- Convergence analysis: Gradient Methods
- Graphical Model & Bayesian Inference
- Bayesian Optimization

Ref:

1. https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-438-algorithms-for-inference-fall-2014/lecture-notes/
2. https://www.cs.cornell.edu/courses/cs4787/2019sp/notes/

Graphical model

- Graphical model describes structures (sparsity, independence, partition) in joint distributions

Bayesian rule

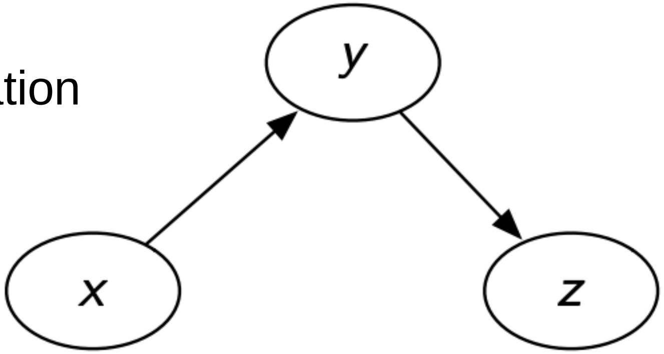
$$p_{x,y,z} = p_{z|x,y}p_{y|x}p_x$$

Graphical model

- Graphical model describes structures (sparsity, independence, partition) in joint distributions

Bayesian rule

Structural information

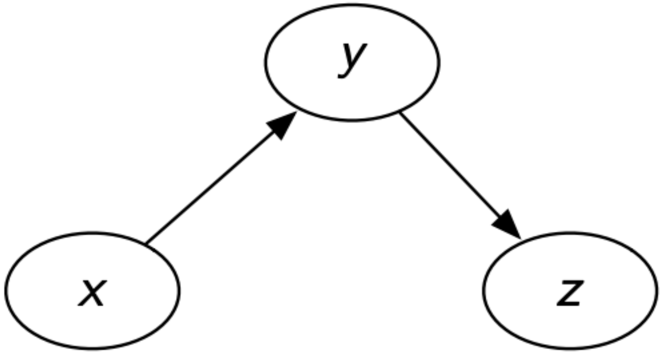


$$p_{x,y,z} = p_{z|x,y} p_{y|x} p_x \qquad p_{x,y,z} = p_{z|y} p_{y|x} p_x.$$

Graphical model

- Graphical model describes structures (sparsity, independence, partition) in joint distributions

Bayesian rule



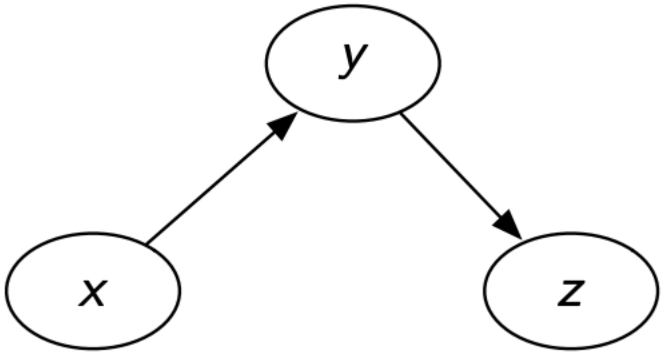
$$p_{x,y,z} = p_{z|\boxed{x},y}\,p_{y|x}\,p_x$$

$$p_{x,y,z} = p_{z|y}\,p_{y|x}\,p_x.$$

Graphical model

- Graphical model describes structures (sparsity, independence, partition) in joint distributions

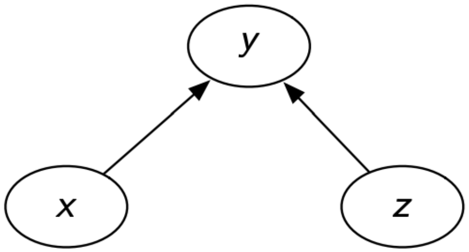
Bayesian rule



$$p_{x,y,z} = p_{z|\boxed{x},y}\, p_{y|x}\, p_x \qquad p_{x,y,z} = p_{z|y}\, p_{y|x}\, p_x.$$

Graphical model: DAG

- Directed acyclic graph



$$p_{x,y,z} = p_x p_{y|x,z} p_z.$$

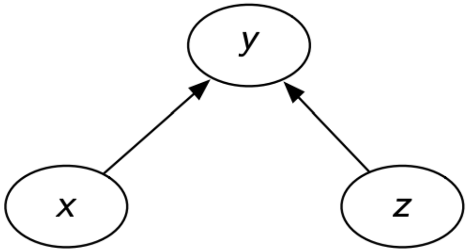
Graphical model: DAG

- Directed acyclic graph

$$\sum_{x_i \in \mathcal{X}} f_i(x_i, x_{\pi_i}) = 1,$$

$$\prod_i f_i(x_i, x_{\pi_i}) = p(x_1, \ldots, x_n),$$

Set of parent nodes



$$p_{x,y,z} = p_x p_{y|x,z} p_z.$$

Graphical model: DAG

- Directed acyclic graph

$$\sum_{x_i \in \mathcal{X}} f_i(x_i, x_{\pi_i}) = 1,$$

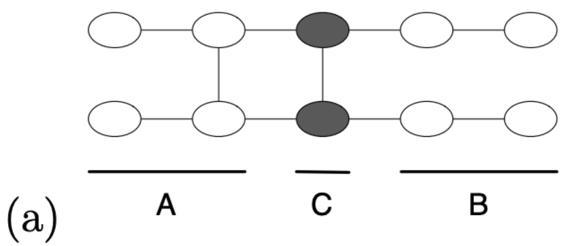
$$\prod_i f_i(x_i, x_{\pi_i}) = p(x_1, \ldots, x_n),$$

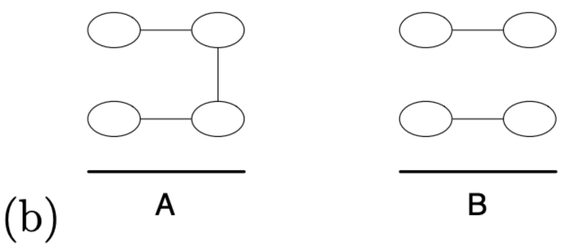
Set of parent nodes

$$\Rightarrow \quad f_i(x_i, x_{\pi_i}) = p_{x_i|x_{\pi_i}}(x_i|x_{\pi_i})$$

Graphical model: Undirected graph

- Undirected graph: Independence
  - $x_A \perp\!\!\!\perp x_B | x_C$ whenever there is no path from a node in A to a node in B which does not pass through a node in C.





Graphical model: Undirected graph

- Undirected graph: Independence
  - $x_A \perp\!\!\!\perp x_B | x_C$ whenever there is no path from a node in A to a node in B which does not pass through a node in C.

$$p(x) = \frac{1}{Z} \prod_{c \in C} \psi_c(x_c)$$

Maximal fully connected subgraphs
(e.g. a pair of nodes)

Graphical model: Undirected graph

- Undirected graph
  - $x_A \perp\!\!\!\perp x_B | x_C$ whenever there is no path from a node in A to a node in B which does not pass through a node in C.

$$p(x) = \frac{1}{Z} \prod_{c \in C} \psi_c(x_c)$$

Maximal fully connected subgraphs
(e.g. a pair of nodes)

Graphical model: Undirected graph

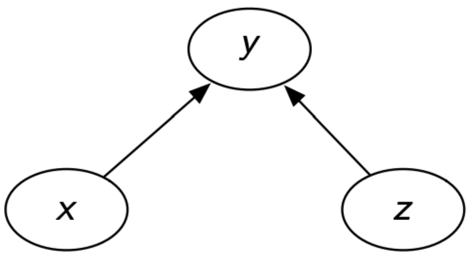
- Undirected graph
  - $x_A \perp\!\!\!\perp x_B | x_C$ whenever there is no path from a node in A to a node in B which does not pass through a node in C.

$$p(x) = \frac{1}{Z} \prod_{c \in C} \psi_c(x_c)$$

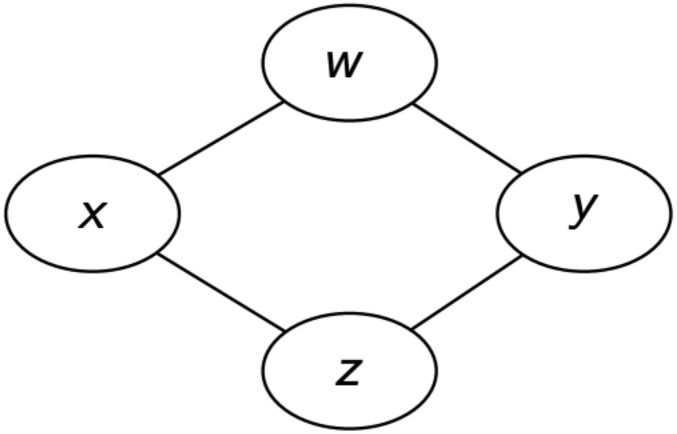
Maximal fully connected subgraphs
(e.g. a pair of nodes)

Graphical model

- Directed acyclic graph vs Undirected graph
  - They do not describe the same set of independent relations.

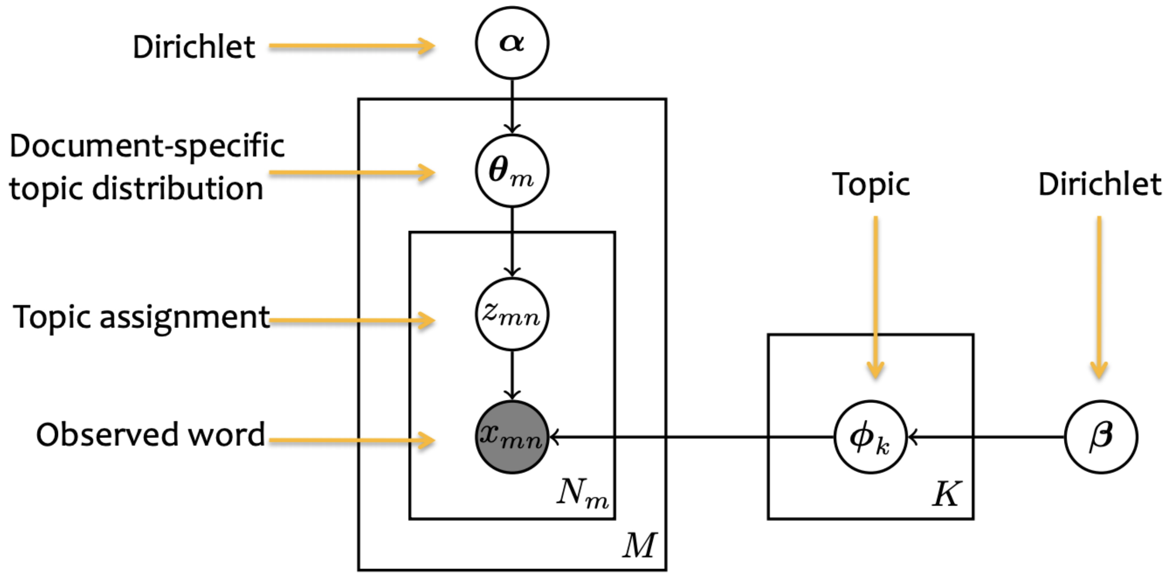


$$p_{x,y,z} = p_x p_{y|x,z} p_z.$$



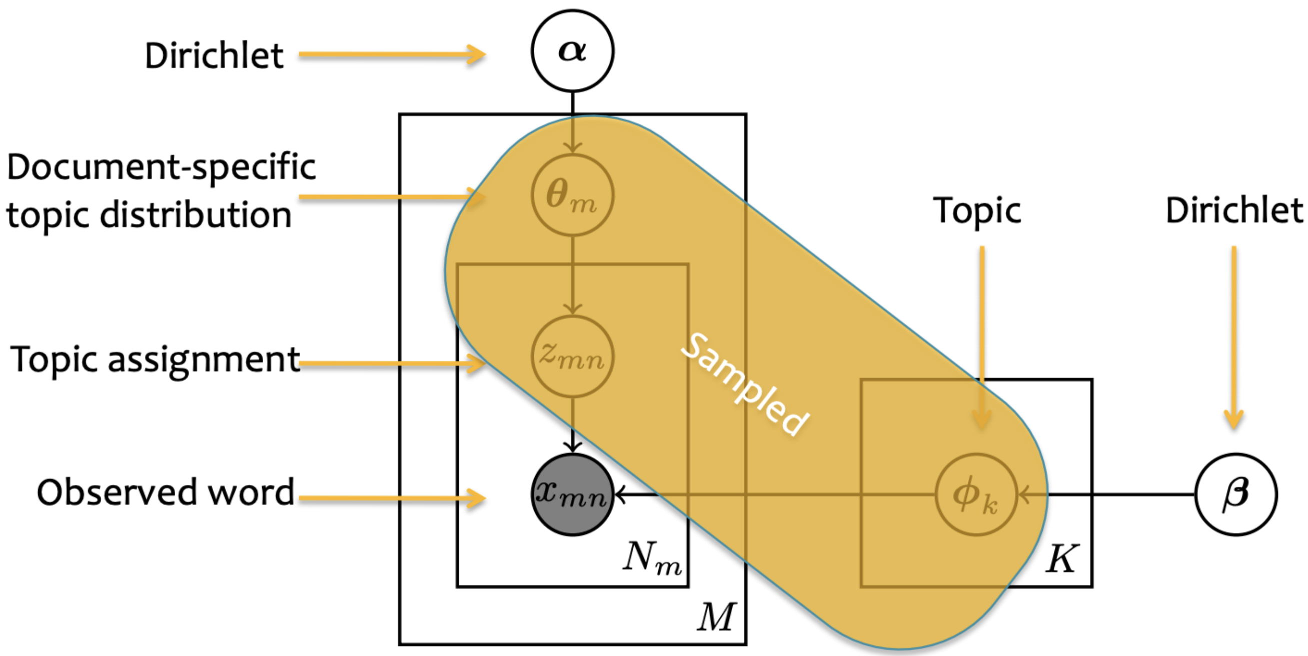
Graphical model : Applications

- Deduce latent variables
  - What is the topic (latent variable theta) of an essay (observed variable W)?



Inference means being able to describe and sample the latent variable of interest given observations. (e.g. Find the topic distribution of a document with its observed words)

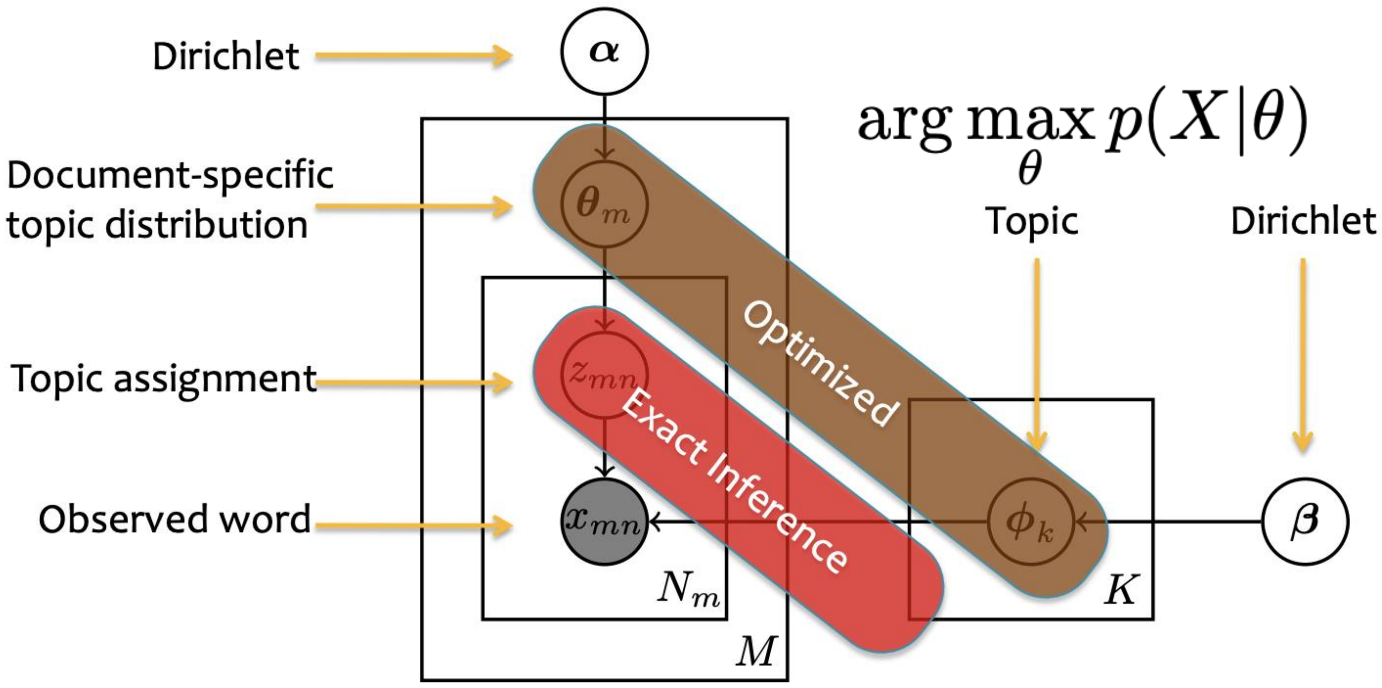
Infer latent variables: MCMC Algorithm



MCMC algorithm: Gibbs sampling

- Suppose we want sample from $p(x, y, z)$
- At iteration t:
  - Sample x_{t+1} from $p(\ \ , y_t , z_t)$
  - Sample y_{t+1} from $p(x_{t+1},\ \ , z_t)$
  - Sample z_{t+1} from $p(x_{t+1},\ y_{t+1},\ \ )$
  - Repeat
- Under certain assumptions, the distribution converges to $p(x, y, z)$

Infer latent variables: Variational Inference



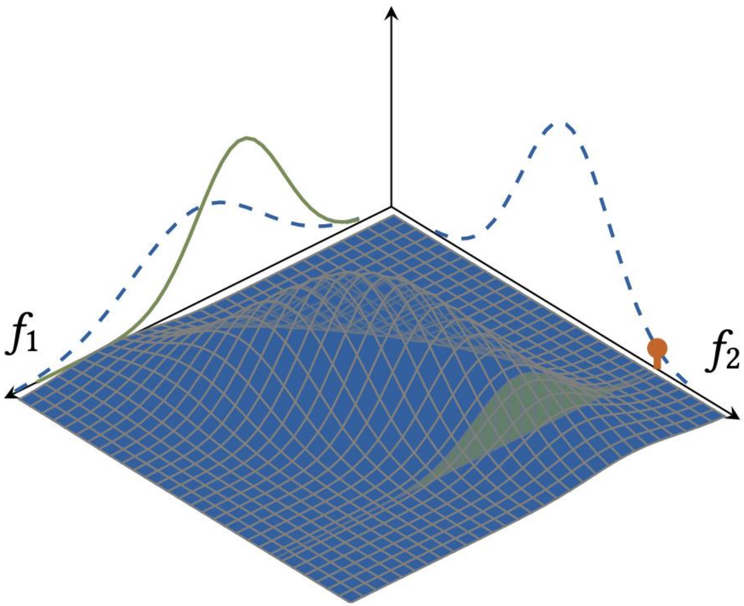
http://www.cs.cmu.edu/~mgormley/courses/10418/slides/lecture21-variational.pdf

Outline

- Optimization Algorithms
- Convergence analysis: Gradient Methods
- Graphical Model & Bayesian Inference
- Bayesian Optimization
    - http://www.it.uu.se/edu/course/homepage/apml/lectures/lecture7_handout.pdf

Multivariate Gaussian:

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_n^2 \end{bmatrix} \right)$$



Infinite dimensional gaussian

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \dots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & \dots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \dots & \sigma_n^2 \end{bmatrix} \right)$$

How should we describe the set of random variables?

Infinite dimensional gaussian

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \dots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & \dots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \dots & \sigma_n^2 \end{bmatrix} \right)$$

How should we describe an infinite set of random variables?

Infinite dimensional gaussian

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \dots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & \dots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \dots & \sigma_n^2 \end{bmatrix} \right)$$

How should we describe an infinite set of random variables?
Use a function to describe the covariance.

$$\begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \kappa(x_1, x_1) & \kappa(x_1, x_2) & \dots & \kappa(x_1, x_n) \\ \kappa(x_2, x_1) & \kappa(x_2, x_2) & \dots & \kappa(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(x_n, x_1) & \kappa(x_n, x_2) & \dots & \kappa(x_n, x_n) \end{bmatrix} \right)$$

Gaussian process: Infinite dimensional Gaussian with a structure on the covariance.

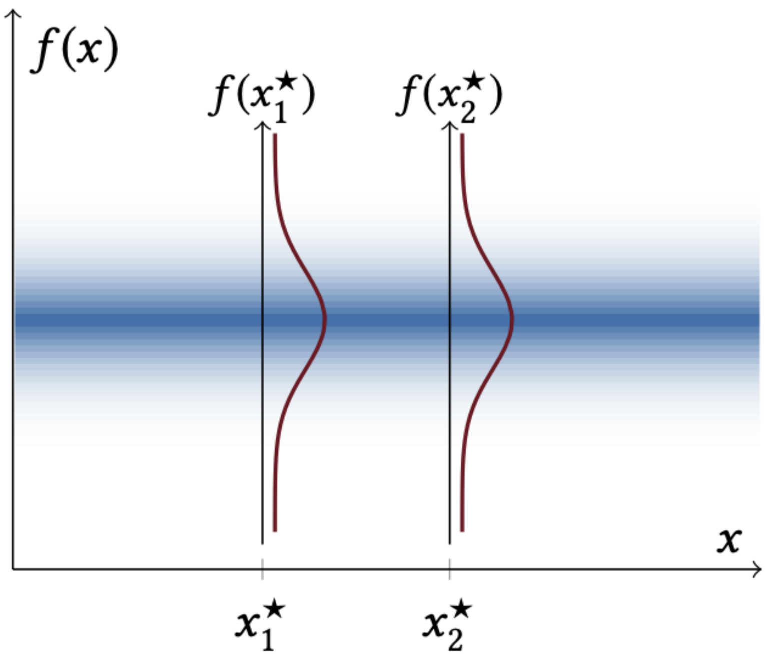
$$
\begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \kappa(x_1, x_1) & \kappa(x_1, x_2) & \dots & \kappa(x_1, x_n) \\ \kappa(x_2, x_1) & \kappa(x_2, x_2) & \dots & \kappa(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(x_n, x_1) & \kappa(x_n, x_2) & \dots & \kappa(x_n, x_n) \end{bmatrix} \right)
$$

$\kappa(x, x')$ needs to be a kernel function (symmetric, positive).

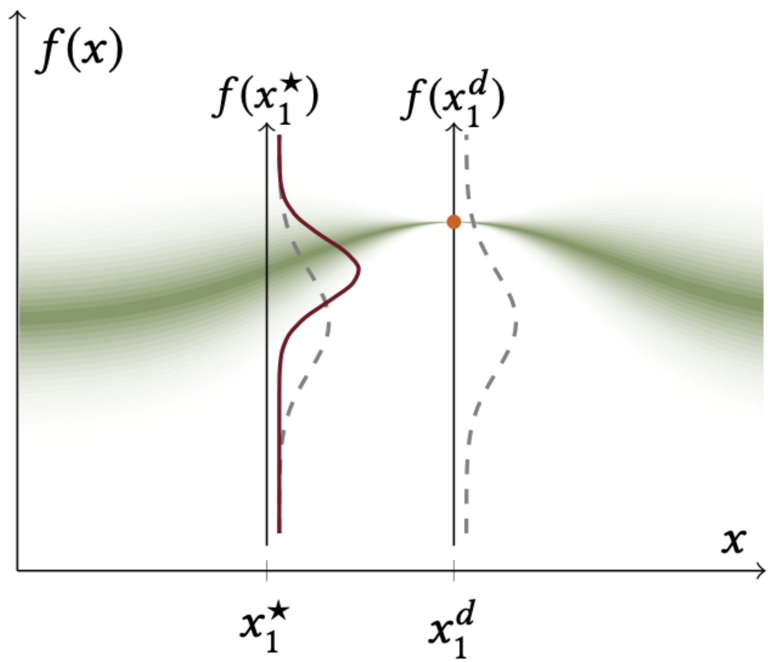
Example:

$$
\kappa(x, x') = \left(1 + \frac{|x - x'|^2}{2\alpha\ell}\right)^{-\alpha},
$$

Gaussian process allows information propagation



The distribution for $f(x^\star)$ without any observations

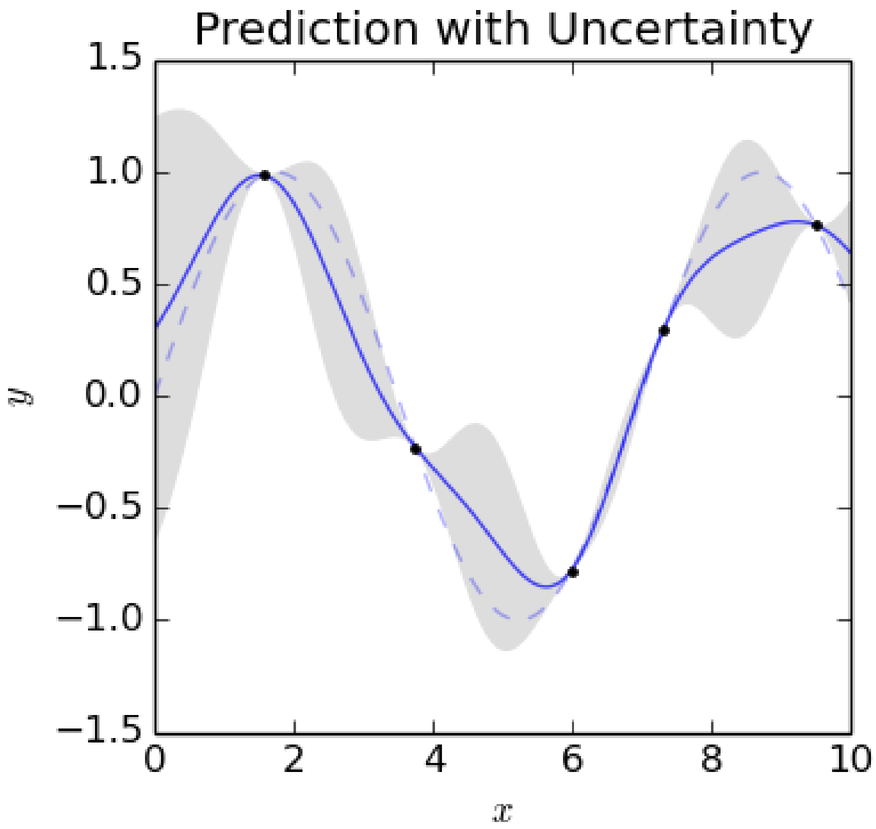


The distribution for $f(x^\star)$ conditional on an observation of x_1^d

Prior

Posterior

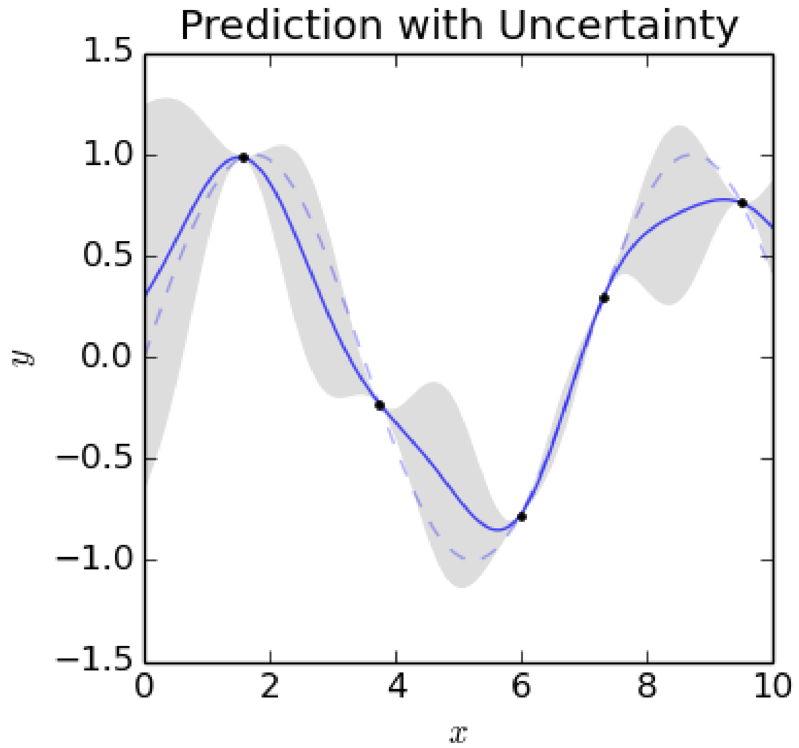
Iterative belief (posterior distribution) update



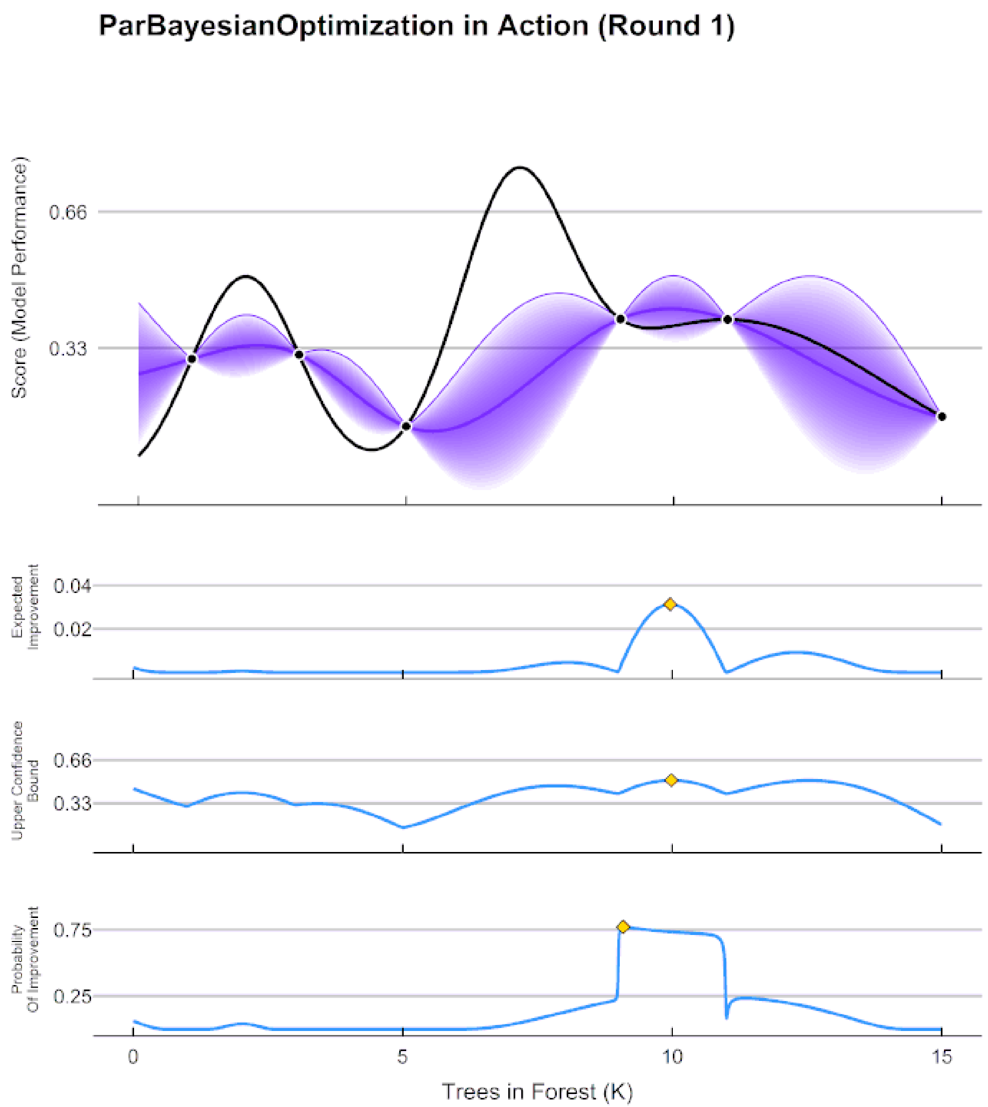
https://en.wikipedia.org/wiki/Gaussian_process

Bayesian Optimization

- Setup:
  - Assume f(x) is sampled from a Gaussian process.
  - We want to find x that maximizes f(x) using only limited function value calls.



Bayesian Optimization: Intuition



Bayesian Optimization: Intuition

Algorithm 1 Bayesian optimization with Gaussian process prior

input: loss function f, kernel K, acquisition function a, loop counts N_{warmup} and N

▷ warmup phase

$y_{\text{best}} \leftarrow \infty$

for $i = 1$ **to** N_{warmup} **do**

  select x_i via some method (usually random sampling)

  compute exact loss function $y_i \leftarrow f(x_i)$

  if $y_i \leq y_{\text{best}}$ **then**

    $x_{\text{best}} \leftarrow x_i$

    $y_{\text{best}} \leftarrow y_i$

  end if

end for

for $i = N_{\text{warmup}} + 1$ **to** N **do**

  update kernel matrix $\Sigma \in \mathbb{R}^{i \times i}$ according to (1)

  let $\mu(x_*)$ and $\sigma(x_*)$ denote the expected value and standard deviation, respectively, of $f(x_*)$ under the Gaussian process model, conditioned on all the previous observations of $f(x_i) = y_i$

  $x_i \leftarrow \arg\min_{x_*} \; a(\mu(x_*), \sigma(x_*), y_{\text{best}})$

  compute exact loss function $y_i \leftarrow f(x_i)$

  if $y_i \leq y_{\text{best}}$ **then**

    $x_{\text{best}} \leftarrow x_i$

    $y_{\text{best}} \leftarrow y_i$

  end if

end for

return x_{best}

Initialize with random samples

Bayesian Optimization: Intuition

Algorithm 1 Bayesian optimization with Gaussian process prior

input: loss function f, kernel K, acquisition function a, loop counts N_{warmup} and N

▷ warmup phase

$y_{\text{best}} \leftarrow \infty$

for $i = 1$ **to** N_{warmup} **do**

  select x_i via some method (usually random sampling)

  compute exact loss function $y_i \leftarrow f(x_i)$

  if $y_i \leq y_{\text{best}}$ **then**

    $x_{\text{best}} \leftarrow x_i$

    $y_{\text{best}} \leftarrow y_i$

  end if

end for

Initialize with random samples

for $i = N_{\text{warmup}} + 1$ **to** N **do**

  update kernel matrix $\Sigma \in \mathbb{R}^{i \times i}$ according to (1)

  let $\mu(x_*)$ and $\sigma(x_*)$ denote the expected value and standard deviation, respectively, of $f(x_*)$ under the Gaussian process model, conditioned on all the previous observations of $f(x_i) = y_i$

Compute posterior

  $x_i \leftarrow \arg\min_{x_*} a(\mu(x_*), \sigma(x_*), y_{\text{best}})$

  compute exact loss function $y_i \leftarrow f(x_i)$

  if $y_i \leq y_{\text{best}}$ **then**

    $x_{\text{best}} \leftarrow x_i$

    $y_{\text{best}} \leftarrow y_i$

  end if

end for

return x_{best}

Bayesian Optimization: Intuition

Algorithm 1 Bayesian optimization with Gaussian process prior

input: loss function f, kernel K, acquisition function a, loop counts N_{warmup} and N

▷ warmup phase

$y_{\text{best}} \leftarrow \infty$

for $i = 1$ **to** N_{warmup} **do**

  select x_i via some method (usually random sampling)

  compute exact loss function $y_i \leftarrow f(x_i)$

  if $y_i \leq y_{\text{best}}$ **then**

    $x_{\text{best}} \leftarrow x_i$

    $y_{\text{best}} \leftarrow y_i$

  end if

end for

Initialize with random samples

for $i = N_{\text{warmup}} + 1$ **to** N **do**

  update kernel matrix $\Sigma \in \mathbb{R}^{i \times i}$ according to (1)

  let $\mu(x_*)$ and $\sigma(x_*)$ denote the expected value and standard deviation, respectively, of $f(x_*)$ under the Gaussian process model, conditioned on all the previous observations of $f(x_i) = y_i$

Compute posterior

  $x_i \leftarrow \arg\min_{x_*} a(\mu(x_*), \sigma(x_*), y_{\text{best}})$

  compute exact loss function $y_i \leftarrow f(x_i)$

  if $y_i \leq y_{\text{best}}$ **then**

    $x_{\text{best}} \leftarrow x_i$

    $y_{\text{best}} \leftarrow y_i$

  end if

end for

return x_{best}

Query the most promising point

Bayesian Optimization: Activation functions

- How do we select the most promising point?

$$x_i \leftarrow \arg\min_{x_*} \; a(\mu(x_*), \sigma(x_*), y_{\text{best}})$$

Bayesian Optimization: Activation functions

- How do we select the most promising point?

$$x_i \leftarrow \arg\min_{x_*} \; a(\mu(x_*), \sigma(x_*), y_{\text{best}})$$

- Probability of improvement

$$a_{\text{PI}}(y_{\text{best}}, \mu, \sigma) = -\Phi\left(\frac{y_{\text{best}} - \mu}{\sigma}\right)$$

Bayesian Optimization: Activation functions

- How do we select the most promising point?

$$x_i \leftarrow \arg\min_{x_*}\ a(\mu(x_*), \sigma(x_*), y_{\text{best}})$$

- Probability of improvement

$$a_{\text{PI}}(y_{\text{best}}, \mu, \sigma) = -\Phi\left(\frac{y_{\text{best}} - \mu}{\sigma}\right)$$

- Expected improvement

$$a_{\text{EI}}(y_{\text{best}}, \mu, \sigma) = -\left(\phi\left(\frac{y_{\text{best}} - \mu}{\sigma}\right) + \frac{y_{\text{best}} - \mu}{\sigma} \cdot \Phi\left(\frac{y_{\text{best}} - \mu}{\sigma}\right)\right) \cdot \sigma.$$

Bayesian Optimization: Activation functions

- How do we select the most promising point?

$$x_i \leftarrow \arg\min_{x_*} \; a(\mu(x_*), \sigma(x_*), y_{\text{best}})$$

- Probability of improvement

$$a_{\text{PI}}(y_{\text{best}}, \mu, \sigma) = -\Phi\left(\frac{y_{\text{best}} - \mu}{\sigma}\right)$$

- Expected improvement

$$a_{\text{EI}}(y_{\text{best}}, \mu, \sigma) = -\left(\phi\left(\frac{y_{\text{best}} - \mu}{\sigma}\right) + \frac{y_{\text{best}} - \mu}{\sigma} \cdot \Phi\left(\frac{y_{\text{best}} - \mu}{\sigma}\right)\right) \cdot \sigma.$$

- Lower confidence

$$a_{\text{LCB}}(y_{\text{best}}, \mu, \sigma) = \mu - \kappa \cdot \sigma.$$

Bayesian optimization: subprocedure

- We need to solve an optimization per step

$$x_i \leftarrow \arg\min_{x_*} \ a(\mu(x_*), \sigma(x_*), y_{\text{best}})$$

- This is usually done with gradient descent and repeated initialization.

Theoretical aspect

- Given the existence of a subprocedure, the complexity of Bayesian optimization is twofold:
  - Sample complexity (number of queries)
  - Computation complexity (number of queries x subprocedure cost)

Theoretical aspect

- Given the existence of a subprocedure, the complexity of Bayesian optimization is twofold:
  - Sample complexity (number of queries)
  - Computation complexity (number of queries x subprocedure cost)

- The sample complexity suffers curse of dimension in the worst case.

- Bayesian optimization is a popular method, but its theoretical advantage still remains to be explained.

Outline

- Optimization Algorithms
- Convergence analysis: Gradient Methods
- Graphical Model & Bayesian Inference
- Bayesian Optimization