GPyTorch

Ludwig Winkler

Machine Learning Group
TU Berlin

May 14, 2019

Outline

Gram-Schmidt Orthogonalization

2^{nd}-Order Optimization of Quadratic Functions

Conjugate Gradient Descent

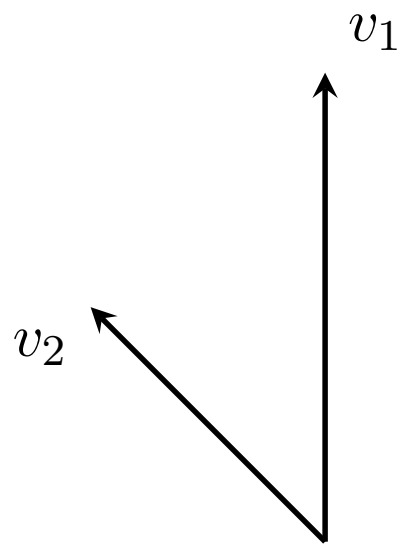
Gaussian Processes

GPyTorch

Going Deep

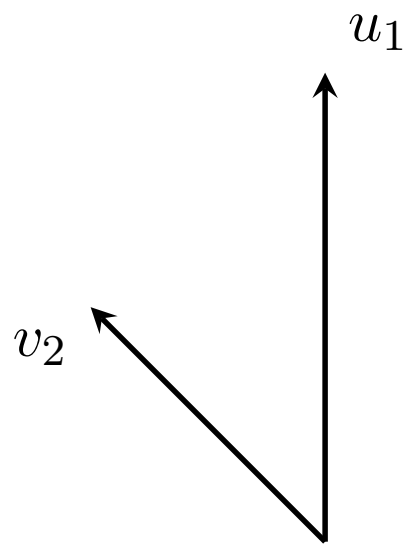
Gram-Schmidt Orthogonolization

- Linearly independent vectors v_1, v_2 span a vector space
- Find orthogonal vectors u_1, u_2 in that vector space



Gram-Schmidt Orthogonolization

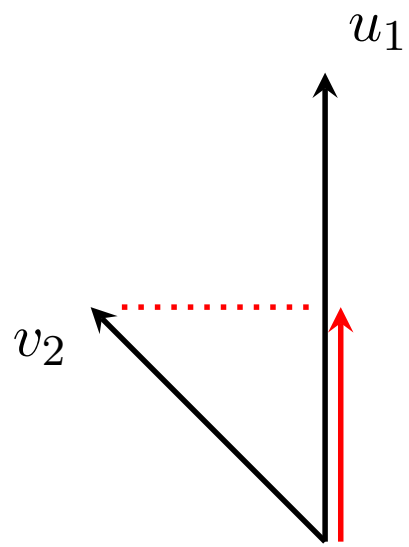
- Linearly independent vectors v_1, v_2 span a vector space
- Find orthogonal vectors u_1, u_2 in that vector space



Gram-Schmidt Orthogonolization

- Linearly independent vectors v_1, v_2 span a vector space
- Find orthogonal vectors u_1, u_2 in that vector space

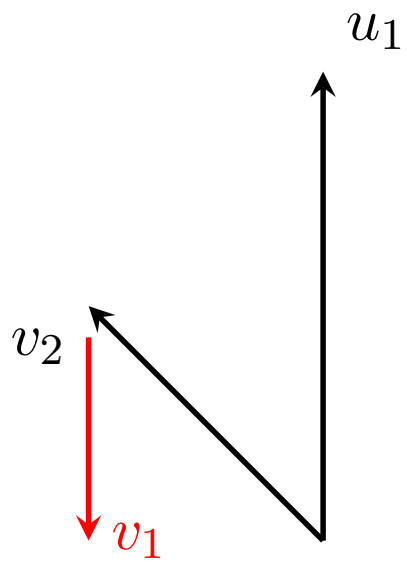
$$\frac{\langle v_2; u_1 \rangle}{\langle u_1; u_1 \rangle} u_1$$



Gram-Schmidt Orthogonolization

- Linearly independent vectors v_1, v_2 span a vector space
- Find orthogonal vectors u_1, u_2 in that vector space

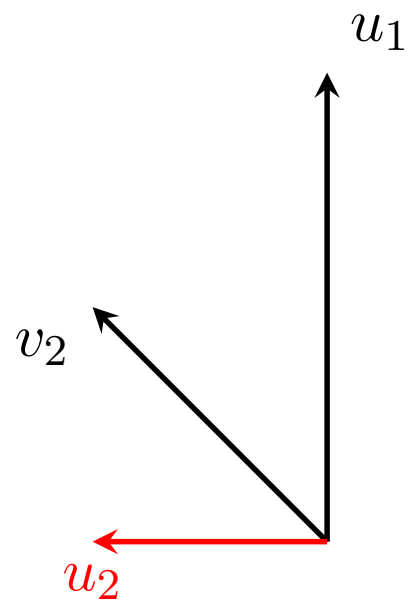
$$v_2 - \frac{\langle v_2; u_1 \rangle}{\langle u_1; u_1 \rangle} u_1$$



Gram-Schmidt Orthogonolization

- Linearly independent vectors v_1, v_2 span a vector space
- Find orthogonal vectors u_1, u_2 in that vector space

$$u_2 = v_2 - \frac{\langle v_2; u_1 \rangle}{\langle u_1; u_1 \rangle} u_1$$



2^{nd}-Order Optimization of Quadratic Functions

- Quadratic optimization problem

$$\min_x \quad f(x) = a(x-d)^2 + e$$

2^{nd}-Order Optimization of Quadratic Functions

- Quadratic optimization problem

$$\min_x \quad f(x) = a(x-d)^2 + e$$

- First and second order derivatives

$$\nabla_x f(x) = 2a(x-d) \quad ; \quad \nabla_x^2 f(x) = 2a$$

2^{nd}-Order Optimization of Quadratic Functions

- Quadratic optimization problem

$$\min_x \quad f(x) = a(x-d)^2 + e$$

- First and second order derivatives

$$\nabla_x f(x) = 2a(x-d) \quad ; \quad \nabla_x^2 f(x) = 2a$$

- Optimal step size due to 2^{nd} order derivative information

$$x^* = x_0 - \frac{\nabla_x f(x_0)}{\nabla_x^2 f(x_0)}$$

2^{nd}-Order Optimization of Quadratic Functions

Taylor Expansion

- 2^{nd}-Order Taylor Expansion around x_0:

$$f(\Delta x) = f(x_0)$$

2^{nd}-Order Optimization of Quadratic Functions

Taylor Expansion

- 2^{nd}-Order Taylor Expansion around x_0:

$$f(\Delta x) = f(x_0) + \nabla_x f(x_0) \Delta x$$

2^{nd}-Order Optimization of Quadratic Functions

Taylor Expansion

- 2^{nd}-Order Taylor Expansion around x_0:

$$f(\Delta x) = f(x_0) + \nabla_x f(x_0)\Delta x + \frac{1}{2}\nabla_x^2 f(x_0)\Delta x^2$$

2^{nd}-Order Optimization of Quadratic Functions

Taylor Expansion

- 2^{nd}-Order Taylor Expansion around x_0:

$$f(\Delta x) = f(x_0) + \nabla_x f(x_0)\Delta x + \frac{1}{2}\nabla_x^2 f(x_0)\Delta x^2$$

- Optimal step size Δx obtained by derivative $\nabla_{\Delta x}$:

$$\nabla_{\Delta x} f(\Delta x) = \nabla_x f(x_0) + \nabla_x^2 f(x_0)\Delta x \stackrel{!}{=} 0$$

2^{nd}-Order Optimization of Quadratic Functions

Taylor Expansion

- 2^{nd}-Order Taylor Expansion around x_0:

$$f(\Delta x) = f(x_0) + \nabla_x f(x_0)\Delta x + \frac{1}{2}\nabla_x^2 f(x_0)\Delta x^2$$

- Optimal step size Δx obtained by derivative $\nabla_{\Delta x}$:

$$\nabla_{\Delta x} f(\Delta x) = \nabla_x f(x_0) + \nabla_x^2 f(x_0)\Delta x \stackrel{!}{=} 0$$

$$\Longleftrightarrow$$

$$\Delta x = -\frac{\nabla_x f(x_0)}{\nabla_x^2 f(x_0)}$$

2^{nd}-Order Optimization of Quadratic Functions

Example

- Quadratic optimization problem with minimum d

$$\min f(x) = a(x-d)^2 + e$$

2^{nd}-Order Optimization of Quadratic Functions

Example

- Quadratic optimization problem with minimum d

$$\min f(x) = a(x-d)^2 + e$$

- First and second order derivatives

$$\nabla_x f(x) = 2a(x-d) \quad ; \quad \nabla_x^2 f(x) = 2a$$

2^{nd}-Order Optimization of Quadratic Functions

Example

- Quadratic optimization problem with minimum d

$$\min f(x) = a(x-d)^2 + e$$

- First and second order derivatives

$$\nabla_x f(x) = 2a(x-d) \quad ; \quad \nabla_x^2 f(x) = 2a$$

- Optimal step size due to 2^{nd} order derivative information

$$x^* = x_0 - \frac{\nabla_x f(x_0)}{\nabla_x^2 f(x_0)}$$

2^{nd}-Order Optimization of Quadratic Functions

Example

- Quadratic optimization problem with minimum d

$$\min f(x) = a(x-d)^2 + e$$

- First and second order derivatives

$$\nabla_x f(x) = 2a(x-d) \quad ; \quad \nabla_x^2 f(x) = 2a$$

- Optimal step size due to 2^{nd} order derivative information

$$\begin{aligned} x^* &= x_0 - \frac{\nabla_x f(x_0)}{\nabla_x^2 f(x_0)} \\ &= x_0 - \frac{2a(x_0-d)}{2a} \end{aligned}$$

2^{nd}-Order Optimization of Quadratic Functions

Example

- Quadratic optimization problem with minimum d

$$\min f(x) = a(x-d)^2 + e$$

- First and second order derivatives

$$\nabla_x f(x) = 2a(x-d) \quad ; \quad \nabla_x^2 f(x) = 2a$$

- Optimal step size due to 2^{nd} order derivative information

$$\begin{aligned} x^* &= x_0 - \frac{\nabla_x f(x_0)}{\nabla_x^2 f(x_0)} \\ &= x_0 - \frac{2a(x_0-d)}{2a} \\ &= x_0 - (x_0 - d) \end{aligned}$$

2^{nd}-Order Optimization of Quadratic Functions

Example

- Quadratic optimization problem with minimum d

$$\min f(x) = a(x-d)^2 + e$$

- First and second order derivatives

$$\nabla_x f(x) = 2a(x-d) \quad ; \quad \nabla_x^2 f(x) = 2a$$

- Optimal step size due to 2^{nd} order derivative information

$$\begin{aligned} x^* &= x_0 - \frac{\nabla_x f(x_0)}{\nabla_x^2 f(x_0)} \\ &= x_0 - \frac{2a(x_0-d)}{2a} \\ &= x_0 - (x_0 - d) \\ &= d \end{aligned}$$

Conjugate Gradient Descent

- Combines conjugate projections with 2^{nd}-Order optimization
- Minimization of quadratic problem with initial point x_0:

$$\min_x \; f(x) = \frac{1}{2} x^T A x + b^T x$$
$$\nabla_x f(x) = Ax + b$$
$$\nabla_x^2 f(x) = A$$

Conjugate Gradient Descent

- Combines conjugate projections with 2^{nd}-Order optimization
- Minimization of quadratic problem with initial point x_0:

$$\min_x \ f(x) = \frac{1}{2} x^T A x + b^T x$$
$$\nabla_x f(x) = Ax + b$$
$$\nabla_x^2 f(x) = A$$

- Notational simplification:

$$g(x) = \nabla_x f(x) = Ax + b$$
$$H = \nabla_x^2 f(x) = A$$

Conjugate Gradient Descent

Search Direction

- Conjugacy includes linear space transformation by A

$$\langle x; y \rangle = x^T y = x^T I y = \langle x; y \rangle_I$$
$$\langle x; y \rangle_A = x^T A y$$

Conjugate Gradient Descent

Search Direction

- Conjugacy includes linear space transformation by A

$$\langle x; y \rangle = x^T y = x^T I y = \langle x; y \rangle_I$$
$$\langle x; y \rangle_A = x^T A y$$

- Initial point x_0 with gradient $g_0 = p_0 = \nabla_x f(x_0)$
- Sequential search directions p_i span vector space

Conjugate Gradient Descent

Search Direction

- Conjugacy includes linear space transformation by A

$$\langle x; y \rangle = x^T y = x^T I y = \langle x; y \rangle_I$$
$$\langle x; y \rangle_A = x^T A y$$

- Initial point x_0 with gradient $g_0 = p_0 = \nabla_x f(x_0)$
- Sequential search directions p_i span vector space
- Enforce orthogonality between sequential search directions

$$p_i = g(x_i) - \sum_{k=0}^{i-1} \frac{\langle g(x_i); p_k \rangle_A}{\langle p_k; p_k \rangle_A} p_k$$

Conjugate Gradient Descent

Search Direction

- Conjugacy includes linear space transformation by A

$$\langle x; y \rangle = x^T y = x^T I y = \langle x; y \rangle_I$$
$$\langle x; y \rangle_A = x^T A y$$

- Initial point x_0 with gradient $g_0 = p_0 = \nabla_x f(x_0)$
- Sequential search directions p_i span vector space
- Enforce orthogonality between sequential search directions

$$p_i = g(x_i) - \sum_{k=0}^{i-1} \frac{\langle g(x_i); p_k \rangle_A}{\langle p_k; p_k \rangle_A} p_k$$

- In N dimensions this leads to a maximum of N search directions

Conjugate Gradient Descent

Step Size

- Initial point x_0 with gradient $p_0 = g(x_0)$
- Taylor Expansion around x_i with $\Delta x_i = x_i - \alpha p_i$:

$$f(\Delta x_i) = f(x_i) + g(x_i)^T(x_i - \Delta x_i) + \frac{1}{2}(x_i - \Delta x_i)^T H(x_i - \Delta x_i)$$

Conjugate Gradient Descent

Step Size

- Initial point x_0 with gradient $p_0 = g(x_0)$
- Taylor Expansion around x_i with $\Delta x_i = x_i - \alpha p_i$:

$$f(\Delta x_i) = f(x_i) + g(x_i)^T \underbrace{(x_i - \Delta x_i)}_{\alpha p_i} + \frac{1}{2} \underbrace{(x_i - \Delta x_i)}_{\alpha p_i}^T H \underbrace{(x_i - \Delta x_i)}_{\alpha p_i}$$

Conjugate Gradient Descent

Step Size

- Initial point x_0 with gradient $p_0 = g(x_0)$
- Taylor Expansion around x_i with $\Delta x_i = x_i - \alpha p_i$:

$$f(\alpha) = f(x_i) + g(x_i)^T(\alpha p_i) + \frac{1}{2}(\alpha p_i)^T H(\alpha p_i)$$

Conjugate Gradient Descent

Step Size

- Initial point x_0 with gradient $p_0 = g(x_0)$
- Taylor Expansion around x_i with $\Delta x_i = x_i - \alpha p_i$:

$$f(\alpha) = f(x_i) + \alpha g(x_i)^T p_i + \frac{1}{2}\alpha^2 p_i^T H p_i$$

Conjugate Gradient Descent

Step Size

- Initial point x_0 with gradient $p_0 = g(x_0)$
- Taylor Expansion around x_i with $\Delta x_i = x_i - \alpha p_i$:

$$f(\alpha) = f(x_i) + \alpha g(x_i)^T p_i + \frac{1}{2}\alpha^2 p_i^T H p_i$$

- Optimal stepsize α computable:

$$\nabla_\alpha f(x_i) = g(x_i)^T p_i + \alpha p_i^T H p_i \stackrel{!}{=} 0$$

Conjugate Gradient Descent

Step Size

- Initial point x_0 with gradient $p_0 = g(x_0)$
- Taylor Expansion around x_i with $\Delta x_i = x_i - \alpha p_i$:

$$f(\alpha) = f(x_i) + \alpha g(x_i)^T p_i + \frac{1}{2}\alpha^2 p_i^T H p_i$$

- Optimal stepsize α computable:

$$\nabla_\alpha f(x_i) = g(x_i)^T p_i + \alpha p_i^T H p_i \stackrel{!}{=} 0$$

$$\alpha = -\frac{g(x_i)^T p_i}{p_i^T H p_i} = -\frac{(Ax_i + b)^T p_i}{p_i^T A p_i}$$

Iterative Conjugate Gradient Descent

- Leverage A-orthogonality of Krylov sequence
- Search direction are mutually orthogonal due to optimal α
- Allows recursive computation of stepsize α and descent correction β

Iterative Conjugate Gradient Descent

- Leverage A-orthogonality of Krylov sequence
- Search direction are mutually orthogonal due to optimal α
- Allows recursive computation of stepsize α and descent correction β

- Error metric $||Ax + b||_2^2 \leq \epsilon$
- Usually $p \ll D$ steps in D dimensions required for convergence
- Linear solve with only Matrix-Vector-Multiplications (MVM)

Gaussian Processes

- Data set $\mathcal{D} = \{X, y\} = \{x_n, y_n\}_{n=0}^{N}$
- Compute kernel matrix $[K_{XX}]_{ij} = K(x_i, x_j)$

$$K(x_i, x_j) = \alpha \exp\left[-\frac{1}{2}\frac{||x_i - x_j||_2}{l^2}\right]$$

$$\theta = \{\alpha, l\}$$

Gaussian Processes

- Data set $\mathcal{D} = \{X, y\} = \{x_n, y_n\}_{n=0}^{N}$
- Compute kernel matrix $[K_{XX}]_{ij} = K(x_i, x_j)$

$$K(x_i, x_j) = \alpha \exp\left[-\frac{1}{2}\frac{||x_i - x_j||_2}{l^2}\right]$$

$$\theta = \{\alpha, l\}$$

- Use K_{XX} as covariance matrix in $\mathcal{N}(y|0, K_{XX})$
- Large entries in K_{XX} result in high covariance in y

Gaussian Processes

Inference in Gaussian Processes

- Compute predictive distribution

$$p(y_*|y, X, X_*) = \mathcal{N}(y_*|\mu(y, X, X_*), \Sigma(y, X, X_*))$$

Gaussian Processes

Inference in Gaussian Processes

- Compute predictive distribution

$$p(y_*|y, X, X_*) = \mathcal{N}(y_*|\mu(y, X, X_*), \Sigma(y, X, X_*))$$

- Mean $\mu(y, X, X_*)$ and covariance $\Sigma(y, X, X_*)$ computable with

$$\mu(y, X, X_*) = K_{X_*X} K_{XX}^{-1} y$$
$$\Sigma(X, X_*) = K_{X_*X_*} - K_{X_*X} K_{XX}^{-1} K_{XX_*}$$

Gaussian Processes

Derivation

- Straightforward derivation of conditional from normal distribution

$$p(y_*, y, X_*, X) \propto \exp\left[-\frac{1}{2}\begin{bmatrix} y \\ y_* \end{bmatrix}^T \begin{bmatrix} K_{XX} & K_{XX_*} \\ K_{X_*X} & K_{X_*X_*} \end{bmatrix}^{-1} \begin{bmatrix} y \\ y_* \end{bmatrix}\right]$$

Gaussian Processes

Derivation

- Straightforward derivation of conditional from normal distribution

$$p(y_*, y, X_*, X) \propto \exp\left[-\frac{1}{2}\begin{bmatrix} y \\ y_* \end{bmatrix}^T \begin{bmatrix} K_{XX} & K_{XX_*} \\ K_{X_*X} & K_{X_*X_*} \end{bmatrix}^{-1} \begin{bmatrix} y \\ y_* \end{bmatrix}\right]$$

$$= \exp\left[-\frac{1}{2}\begin{bmatrix} y \\ y_* \end{bmatrix}^T \begin{bmatrix} P & Q \\ R & S \end{bmatrix} \begin{bmatrix} y \\ y_* \end{bmatrix}\right]$$

Gaussian Processes

Derivation

- Straightforward derivation of conditional from normal distribution

$$
\begin{aligned}
p(y_*, y, X_*, X) &\propto \exp\left[-\frac{1}{2}\begin{bmatrix} y \\ y_* \end{bmatrix}^T \begin{bmatrix} K_{XX} & K_{XX_*} \\ K_{X_*X} & K_{X_*X_*} \end{bmatrix}^{-1} \begin{bmatrix} y \\ y_* \end{bmatrix}\right] \\
&= \exp\left[-\frac{1}{2}\begin{bmatrix} y \\ y_* \end{bmatrix}^T \begin{bmatrix} P & Q \\ R & S \end{bmatrix} \begin{bmatrix} y \\ y_* \end{bmatrix}\right] \\
&= \exp\left[-\frac{1}{2}\left(y^T P y + y^T Q y_* + y_*^T R y + y_*^T S y_*\right)\right]
\end{aligned}
$$

Gaussian Processes

Derivation

- Straightforward derivation of conditional from normal distribution

$$
\begin{aligned}
p(y_*, y, X_*, X) &\propto \exp\left[-\frac{1}{2}\begin{bmatrix} y \\ y_* \end{bmatrix}^T \begin{bmatrix} K_{XX} & K_{XX_*} \\ K_{X_*X} & K_{X_*X_*} \end{bmatrix}^{-1} \begin{bmatrix} y \\ y_* \end{bmatrix}\right] \\
&= \exp\left[-\frac{1}{2}\begin{bmatrix} y \\ y_* \end{bmatrix}^T \begin{bmatrix} P & Q \\ R & S \end{bmatrix} \begin{bmatrix} y \\ y_* \end{bmatrix}\right] \\
&= \exp\left[-\frac{1}{2}\left(y^T P y + y^T Q y_* + y_*^T R y + y_*^T S y_*\right)\right] \\
p(y_*|y, X, X_*) &\propto \exp\left[-\frac{1}{2}\left(y^T Q y_* + y_*^T R y + y_*^T S y_*\right)\right]
\end{aligned}
$$

Gaussian Processes

Training in Gaussian Processes

- Minimize marginal NLL to optimize kernel parameters

$$\min_{\theta} -\log p(\mathcal{D};\theta) \propto \log |K_{XX}| - y^T K_{XX}^{-1} y$$

Gaussian Processes

Training in Gaussian Processes

- Minimize marginal NLL to optimize kernel parameters

$$\min_{\theta} -\log p(\mathcal{D};\theta) \propto \log |K_{XX}| - y^T K_{XX}^{-1} y$$

- Optimize θ with

$$\nabla_\theta -\log p(\mathcal{D};\theta) = \mathsf{Tr}\left[K_{XX}^{-1}\frac{dK_{XX}}{d\theta}\right] + y^T K_{XX}^{-1}\frac{dK_{XX}}{d\theta}K_{XX}^{-1}y$$

Gaussian Processes

Training in Gaussian Processes

- Minimize marginal NLL to optimize kernel parameters

$$\min_{\theta} -\log p(\mathcal{D};\theta) \propto \log|K_{XX}| - y^T K_{XX}^{-1} y$$

- Optimize θ with

$$\nabla_\theta -\log p(\mathcal{D};\theta) = \mathsf{Tr}\left[K_{XX}^{-1}\frac{dK_{XX}}{d\theta}\right] + y^T K_{XX}^{-1}\frac{dK_{XX}}{d\theta}K_{XX}^{-1}y$$

- Linear algebra becomes prohibitive for large sample sizes N
- Learn smaller data set $\mathcal{D}^{IP} = \{\overline{x}_n, \overline{y}_n\}_{n=0}^{N^{IP}}$

Computational Complexity

Computationally expensive operations:

- $K_{XX}^{-1} y = u$ occurs frequently and is expensive
- $\log |K_{XX}|$ in marginal NLL
- Tr $\left[K_{XX}^{-1} \frac{dK_{XX}}{d\theta} \right]$ in kernel parameter optimization

Computational Complexity

Computationally expensive operations:

- $K_{XX}^{-1} y = u$ occurs frequently and is expensive
- $\log |K_{XX}|$ in marginal NLL
- Tr $\left[K_{XX}^{-1} \frac{dK_{XX}}{d\theta} \right]$ in kernel parameter optimization

Modified Batched Conjugate Gradients (mBCG):

- Conjugate Gradient Descent with Preconditioning
- Hutchinsons Stochastic Trace Estimation
- Tridiagonalization with Lanczos Algorithm

CG with Preconditioning

- Minimize reformulated problem $K_{XX}^{-1}y = u$

$$\min_{u} \; ||K_{XX}^{-1}y - u||$$

CG with Preconditioning

- Minimize reformulated problem $K_{XX}^{-1} y = u$

$$\min_u \ ||K_{XX}^{-1} y - u|| = \min_u ||y - K_{XX} \ u||$$

CG with Preconditioning

- Minimize reformulated problem $K_{XX}^{-1} y = u$

$$\min_u \ ||K_{XX}^{-1} y - u|| = \min_u ||y - K_{XX} \ u||$$

- Use CG to optimize $||y - K_{XX} \ u|| \leq \epsilon$ in $p \ll N$ iterations

CG with Preconditioning

- Minimize reformulated problem $K_{XX}^{-1} y = u$

$$\min_u \ ||K_{XX}^{-1} y - u|| = \min_u ||y - K_{XX} \ u||$$

- Use CG to optimize $||y - K_{XX} \ u|| \leq \epsilon$ in $p \ll N$ iterations
- Precondition $K_{XX} \Rightarrow P K_{XX}$ to reduce pathological curvature

CG with Preconditioning

- Minimize reformulated problem $K_{XX}^{-1} y = u$

$$\min_u \ ||K_{XX}^{-1} y - u|| = \min_u ||y - K_{XX} \ u||$$

- Use CG to optimize $||y - K_{XX} \ u|| \leq \epsilon$ in $p \ll N$ iterations
- Precondition $K_{XX} \Rightarrow P K_{XX}$ to reduce pathological curvature
- Optimal $P = K_{XX}^{-1}$, but low rank approximation already sufficient

CG with Preconditioning

- Minimize reformulated problem $K_{XX}^{-1} y = u$

$$\min_u \ ||K_{XX}^{-1} y - u|| = \min_u ||y - K_{XX} \ u||$$

- Use CG to optimize $||y - K_{XX} \ u|| \leq \epsilon$ in $p \ll N$ iterations
- Precondition $K_{XX} \Rightarrow P K_{XX}$ to reduce pathological curvature
- Optimal $P = K_{XX}^{-1}$, but low rank approximation already sufficient
- Pivoted Cholesky decomposition $K_{XX} = L_k L_k^T$ of rank k

CG with Preconditioning

- Minimize reformulated problem $K_{XX}^{-1} y = u$

$$\min_u \ ||K_{XX}^{-1} y - u|| = \min_u ||y - K_{XX} \ u||$$

- Use CG to optimize $||y - K_{XX} \ u|| \leq \epsilon$ in $p \ll N$ iterations
- Precondition $K_{XX} \Rightarrow P K_{XX}$ to reduce pathological curvature
- Optimal $P = K_{XX}^{-1}$, but low rank approximation already sufficient
- Pivoted Cholesky decomposition $K_{XX} = L_k L_k^T$ of rank k
- Computation, reconstruction, inverse and log determinant all $\mathcal{O}(Nk^2)$

CG with Preconditioning

- Minimize reformulated problem $K_{XX}^{-1} y = u$

$$\min_u \ ||K_{XX}^{-1} y - u|| = \min_u ||y - K_{XX} \ u||$$

- Use CG to optimize $||y - K_{XX} \ u|| \leq \epsilon$ in $p \ll N$ iterations
- Precondition $K_{XX} \Rightarrow P K_{XX}$ to reduce pathological curvature
- Optimal $P = K_{XX}^{-1}$, but low rank approximation already sufficient
- Pivoted Cholesky decomposition $K_{XX} = L_k L_k^T$ of rank k
- Computation, reconstruction, inverse and log determinant all $\mathcal{O}(Nk^2)$
- $k = 5, 7, 9, k \ll N$ already offer significant gains

Stochastic Trace Estimation

- Trace of a square matrix $A \in \mathbb{R}^{N \times N}$: $Tr(A) = \sum_{i=0}^{N} a_{ii}$

Stochastic Trace Estimation

- Trace of a square matrix $A \in \mathbb{R}^{N \times N}$: $Tr(A) = \sum_{i=0}^{N} a_{ii}$
- Random vector $z_i \sim \mathcal{N}(0, I) \in \mathbb{R}^N \Rightarrow \mathbb{E}[zz^T] = I$

Stochastic Trace Estimation

- Trace of a square matrix $A \in \mathbb{R}^{N \times N}$: $Tr(A) = \sum_{i=0}^{N} a_{ii}$
- Random vector $z_i \sim \mathcal{N}(0, I) \in \mathbb{R}^N \Rightarrow \mathbb{E}[zz^T] = I$

$$Tr[A]$$

Stochastic Trace Estimation

- Trace of a square matrix $A \in \mathbb{R}^{N \times N}$: $Tr(A) = \sum_{i=0}^{N} a_{ii}$
- Random vector $z_i \sim \mathcal{N}(0, I) \in \mathbb{R}^N \Rightarrow \mathbb{E}[zz^T] = I$

$$Tr[A] = Tr[A\mathbb{E}[zz^T]]$$

Stochastic Trace Estimation

- Trace of a square matrix $A \in \mathbb{R}^{N \times N}$: $Tr(A) = \sum_{i=0}^{N} a_{ii}$
- Random vector $z_i \sim \mathcal{N}(0, I) \in \mathbb{R}^N \Rightarrow \mathbb{E}[zz^T] = I$

$$Tr[A] = Tr[A\mathbb{E}[zz^T]] = \mathbb{E}[Tr[z^T Az]]$$

Stochastic Trace Estimation

- Trace of a square matrix $A \in \mathbb{R}^{N \times N}$: $Tr(A) = \sum_{i=0}^{N} a_{ii}$
- Random vector $z_i \sim \mathcal{N}(0, I) \in \mathbb{R}^N \Rightarrow \mathbb{E}[zz^T] = I$

$$Tr[A] = Tr[A\mathbb{E}[zz^T]] = \mathbb{E}[Tr[z^T Az]] = \mathbb{E}[z^T Az]$$

Stochastic Trace Estimation

- Trace of a square matrix $A \in \mathbb{R}^{N \times N} : Tr(A) = \sum_{i=0}^{N} a_{ii}$
- Random vector $z_i \sim \mathcal{N}(0, I) \in \mathbb{R}^N \Rightarrow \mathbb{E}[zz^T] = I$

$$Tr[A] = Tr[A\mathbb{E}[zz^T]] = \mathbb{E}[Tr[z^T Az]] = \mathbb{E}[z^T Az]$$

- CG solves iteratively with Matrix-Vector-Multiplications
- Straightforward to parallelize to Matrix-Matrix-Multiplications

$$[y \qquad\qquad\qquad] = K_{XX}[u_0 \qquad\qquad\qquad\qquad]$$

Stochastic Trace Estimation

- Trace of a square matrix $A \in \mathbb{R}^{N \times N}$: $Tr(A) = \sum_{i=0}^{N} a_{ii}$
- Random vector $z_i \sim \mathcal{N}(0, I) \in \mathbb{R}^N \Rightarrow \mathbb{E}[zz^T] = I$

$$Tr[A] = Tr[A\mathbb{E}[zz^T]] = \mathbb{E}[Tr[z^T Az]] = \mathbb{E}[z^T Az]$$

- CG solves iteratively with Matrix-Vector-Multiplications
- Straightforward to parallelize to Matrix-Matrix-Multiplications

$$[y \; z_1 \; z_2 \; \dots z_t] = K_{XX}[u_0 \; u_1 \; u_2 \; \dots \; u_t]$$

Stochastic Trace Estimation

- Trace of a square matrix $A \in \mathbb{R}^{N \times N}$: $Tr(A) = \sum_{i=0}^{N} a_{ii}$
- Random vector $z_i \sim \mathcal{N}(0, I) \in \mathbb{R}^N \Rightarrow \mathbb{E}[zz^T] = I$

$$Tr[A] = Tr[A\mathbb{E}[zz^T]] = \mathbb{E}[Tr[z^T Az]] = \mathbb{E}[z^T Az]$$

- CG solves iteratively with Matrix-Vector-Multiplications
- Straightforward to parallelize to Matrix-Matrix-Multiplications

$$[y \; z_1 \; z_2 \; \dots z_t] = K_{XX}[u_0 \; u_1 \; u_2 \; \dots \; u_t]$$

- $z_i \in \mathbb{R}^N$ and $u_i \in \mathbb{R}^N$ for subsequent stochastic approximations

Modified Batch Conjugate Gradient Descent

- Batch Conjugate Gradient yields $\{u_t = K_{XX}^{-1} z_t\}_{t=0}^{T}$
- Stochastic trace estimation with results from mBCG

$$\text{Tr}\left[K_{XX}^{-1} \frac{dK_{XX}}{d\theta}\right] = \mathbb{E}[\underbrace{z^T K_{XX}^{-1}}_{u} \frac{dK_{XX}}{d\theta} z]$$

Conjugate Gradients and Lanczos

- Eigendecomposition of K_{XX} in $\log\det K_{XX}$

$$\log\det K_{XX} = Tr(\log K_{XX})$$

Conjugate Gradients and Lanczos

- Eigendecomposition of K_{XX} in $\log\det K_{XX}$

$$\log\det K_{XX} = Tr(\log K_{XX}) = Tr(\log(QTQ^T))$$

Conjugate Gradients and Lanczos

- Eigendecomposition of K_{XX} in $\log\det K_{XX}$

$$\log\det K_{XX} = Tr(\log K_{XX}) = Tr(\log(QTQ^T)) = Tr(Q\log TQ^T)$$

Conjugate Gradients and Lanczos

- Eigendecomposition of K_{XX} in $\log\det K_{XX}$

$$\log\det K_{XX} = Tr(\log K_{XX}) = Tr(\log(QTQ^T)) = Tr(Q\log TQ^T)$$

- Lanczos Algorithm closely related to Conjugate Gradient Descent

Conjugate Gradients and Lanczos

- Eigendecomposition of K_{XX} in $\log\det K_{XX}$

$$\log\det K_{XX} = Tr(\log K_{XX}) = Tr(\log(QTQ^T)) = Tr(Q\log TQ^T)$$

- Lanczos Algorithm closely related to Conjugate Gradient Descent
- α's and β's from CG used to construct tridiagonal Lanczos matrix T

Conjugate Gradients and Lanczos

- Eigendecomposition of K_{XX} in $\log\det K_{XX}$

$$\log\det K_{XX} = Tr(\log K_{XX}) = Tr(\log(QTQ^T)) = Tr(Q\log TQ^T)$$

- Lanczos Algorithm closely related to Conjugate Gradient Descent
- α's and β's from CG used to construct tridiagonal Lanczos matrix T
- First column of $Q_t \in \mathbb{R}^{n\times p}$ is $z_t \sim \mathcal{N}(0, I)$

Conjugate Gradients and Lanczos

- Eigendecomposition of K_{XX} in $\log\det K_{XX}$

$$\log\det K_{XX} = Tr(\log K_{XX}) = Tr(\log(QTQ^T)) = Tr(Q\log TQ^T)$$

- Lanczos Algorithm closely related to Conjugate Gradient Descent
- α's and β's from CG used to construct tridiagonal Lanczos matrix T
- First column of $Q_t \in \mathbb{R}^{n\times p}$ is $z_t \sim \mathcal{N}(0, I)$
- Factorizations $\{K_{XX} = Q_t T_t Q_t^T\}_{t=0}^{T}$ as a byproduct of BCG

Conjugate Gradients and Lanczos

- Eigendecomposition of K_{XX} in $\log\det K_{XX}$

$$\log\det K_{XX} = Tr(\log K_{XX}) = Tr(\log(QTQ^T)) = Tr(Q\log TQ^T)$$

- Lanczos Algorithm closely related to Conjugate Gradient Descent
- α's and β's from CG used to construct tridiagonal Lanczos matrix T
- First column of $Q_t \in \mathbb{R}^{n\times p}$ is $z_t \sim \mathcal{N}(0, I)$
- Factorizations $\{K_{XX} = Q_t T_t Q_t^T\}_{t=0}^T$ as a byproduct of BCG
- Reuse stochastic trace estimation trick:

$$Tr(Q\log TQ^T) = \mathbb{E}[z^T Q_t \log T_t Q_t^T z] = \mathbb{E}[e_1^T \log T_t e_1]$$

Recap

- Precondition with low rank approximation $P^{-1} = (L_k L_k^T + \sigma I)^{-1}$
- $u_0 = K_{XX}^{-1} y$ and $\{u_t = K_{XX}^{-1} z_t\}_{t=0}^{T}$ from mBCG
- $\log |K_{XX}| = \mathsf{Tr}\left[\log K_{XX}\right] = \mathbb{E}[e_1^T \log T e_1]$

- Modified training of Gaussian Process:

$$L(\theta | X, y) \propto \log |K_{XX}| + y^T K_{XX}^{-1} y$$

Recap

- Precondition with low rank approximation $P^{-1} = (L_k L_k^T + \sigma I)^{-1}$
- $u_0 = K_{XX}^{-1} y$ and $\{u_t = K_{XX}^{-1} z_t\}_{t=0}^{T}$ from mBCG
- $\log |K_{XX}| = \mathsf{Tr}\left[\log K_{XX}\right] = \mathbb{E}[e_1^T \log T e_1]$

- Modified training of Gaussian Process:

$$L(\theta|X, y) \propto \log |K_{XX}| + y^T K_{XX}^{-1} y$$

$$\Downarrow$$

$$L(\theta|X, y) \propto \mathbb{E}\left[e_1^T \log T e_1\right] + \; y^T u_0$$

Recap

- Precondition with low rank approximation $P^{-1} = (L_k L_k^T + \sigma I)^{-1}$
- $u_0 = K_{XX}^{-1} y$ and $\{u_t = K_{XX}^{-1} z_t\}_{t=0}^{T}$ from mBCG
- $\log |K_{XX}| = \mathsf{Tr}\left[\log K_{XX}\right] = \mathbb{E}[e_1^T \log T e_1]$

- Modified training of Gaussian Process:

$$L(\theta|X, y) \propto \log |K_{XX}| + y^T K_{XX}^{-1} y$$

$$\frac{dL}{d\theta} = y^T K_{XX}^{-1} \frac{dK_{XX}}{d\theta} K_{XX}^{-1} y + \mathsf{Tr}\left[K_{XX}^{-1} \frac{dK_{XX}}{d\theta}\right]$$

$$\Downarrow$$

$$L(\theta|X, y) \propto \mathbb{E}\left[e_1^T \log T e_1\right] + \; y^T u_0$$

Recap

- Precondition with low rank approximation $P^{-1} = (L_k L_k^T + \sigma I)^{-1}$
- $u_0 = K_{XX}^{-1} y$ and $\{u_t = K_{XX}^{-1} z_t\}_{t=0}^{T}$ from mBCG
- $\log |K_{XX}| = \mathsf{Tr}\left[\log K_{XX}\right] = \mathbb{E}[e_1^T \log T e_1]$

- Modified training of Gaussian Process:

$$L(\theta|X,y) \propto \log |K_{XX}| + y^T K_{XX}^{-1} y$$

$$\frac{dL}{d\theta} = y^T K_{XX}^{-1} \frac{dK_{XX}}{d\theta} K_{XX}^{-1} y + \mathsf{Tr}\left[K_{XX}^{-1} \frac{dK_{XX}}{d\theta}\right]$$

$$\Downarrow$$

$$L(\theta|X,y) \propto \mathbb{E}\left[e_1^T \log T e_1\right] + \; y^T u_0$$

$$\frac{dL}{d\theta} = u_0^T \frac{dK_{XX}}{d\theta} u_0 + \mathbb{E}\left[u^T \frac{dK_{XX}}{d\theta} z\right]$$

Recap

- Precondition with low rank approximation $P^{-1} = (L_k L_k^T + \sigma I)^{-1}$
- $u_0 = K_{XX}^{-1} y$ and $\{u_t = K_{XX}^{-1} z_t\}_{t=0}^{T}$ from mBCG
- $\log |K_{XX}| = \mathsf{Tr}\left[\log K_{XX}\right] = \mathbb{E}[e_1^T \log T e_1]$

- Modified training of Gaussian Process:

$$L(\theta|X,y) \propto \log |K_{XX}| + y^T K_{XX}^{-1} y$$

$$\frac{dL}{d\theta} = y^T K_{XX}^{-1} \frac{dK_{XX}}{d\theta} K_{XX}^{-1} y + \mathsf{Tr}\left[K_{XX}^{-1} \frac{dK_{XX}}{d\theta}\right]$$

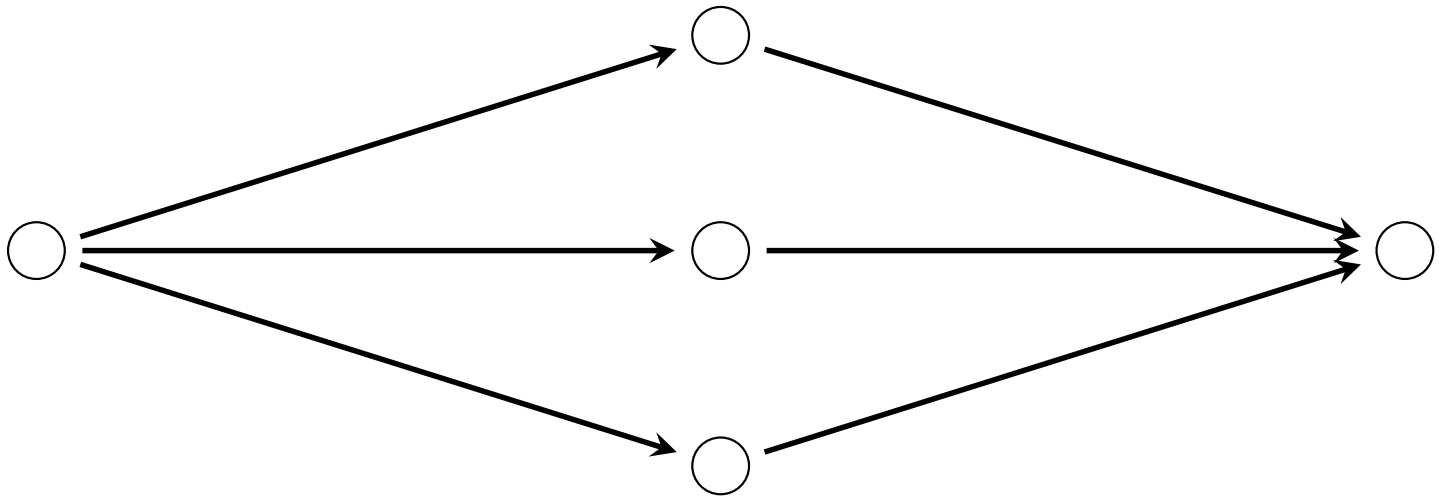
$$\Downarrow$$

$$L(\theta|X,y) \propto \mathbb{E}\left[e_1^T \log T e_1\right] + \; y^T u_0$$

$$\frac{dL}{d\theta} = u_0^T \frac{dK_{XX}}{d\theta} u_0 + \mathbb{E}\left[u^T \frac{dK_{XX}}{d\theta} z\right]$$

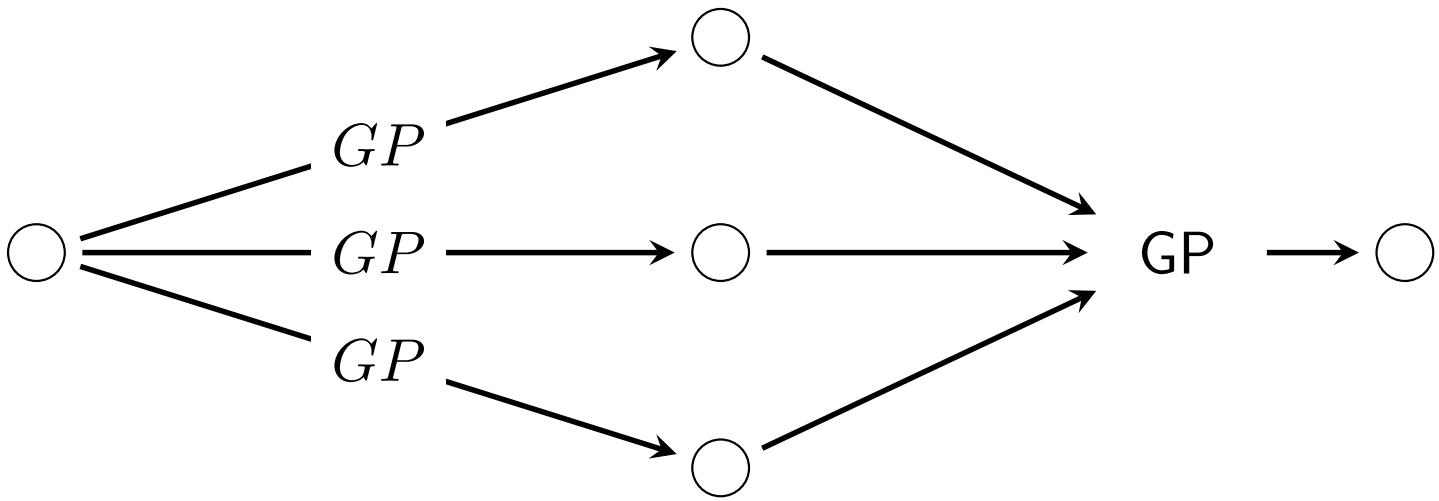
Excursion into Deep Gaussian Processes

- Deep architectures compute layer-wise representations
- Single kernel parameters θ often insufficient



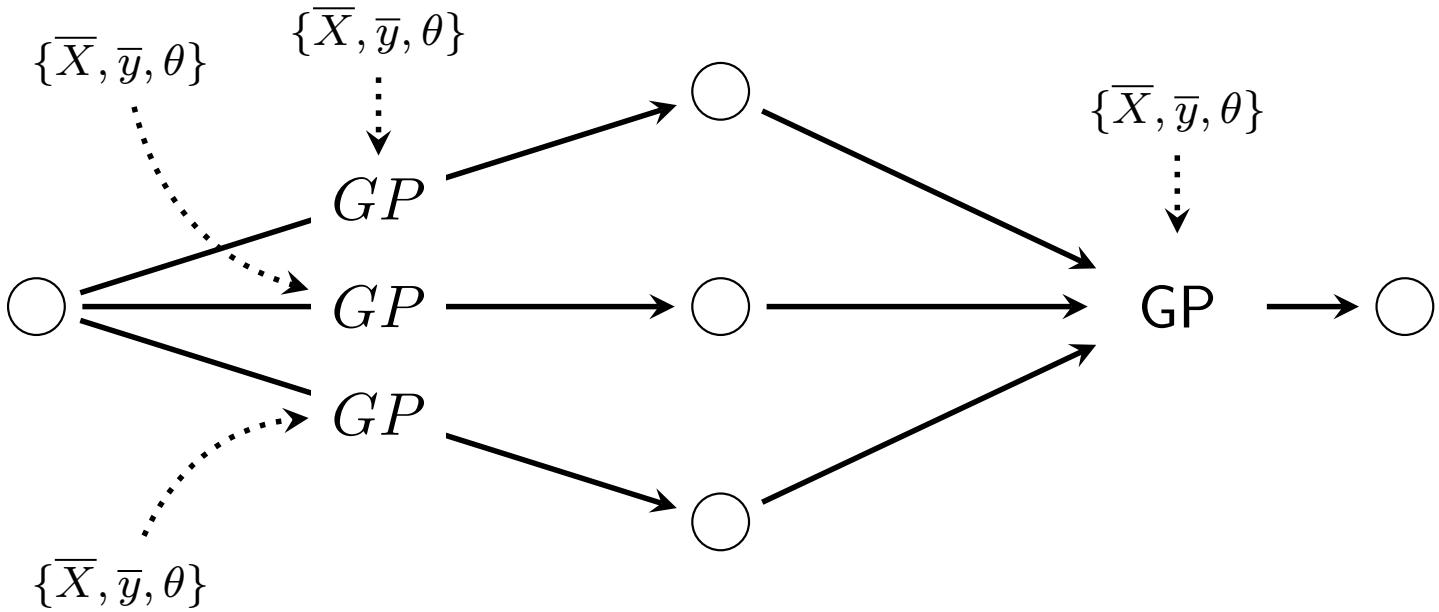
Excursion into Deep Gaussian Processes

- Deep architectures compute layer-wise representations
- Single kernel parameters θ often insufficient



Excursion into Deep Gaussian Processes

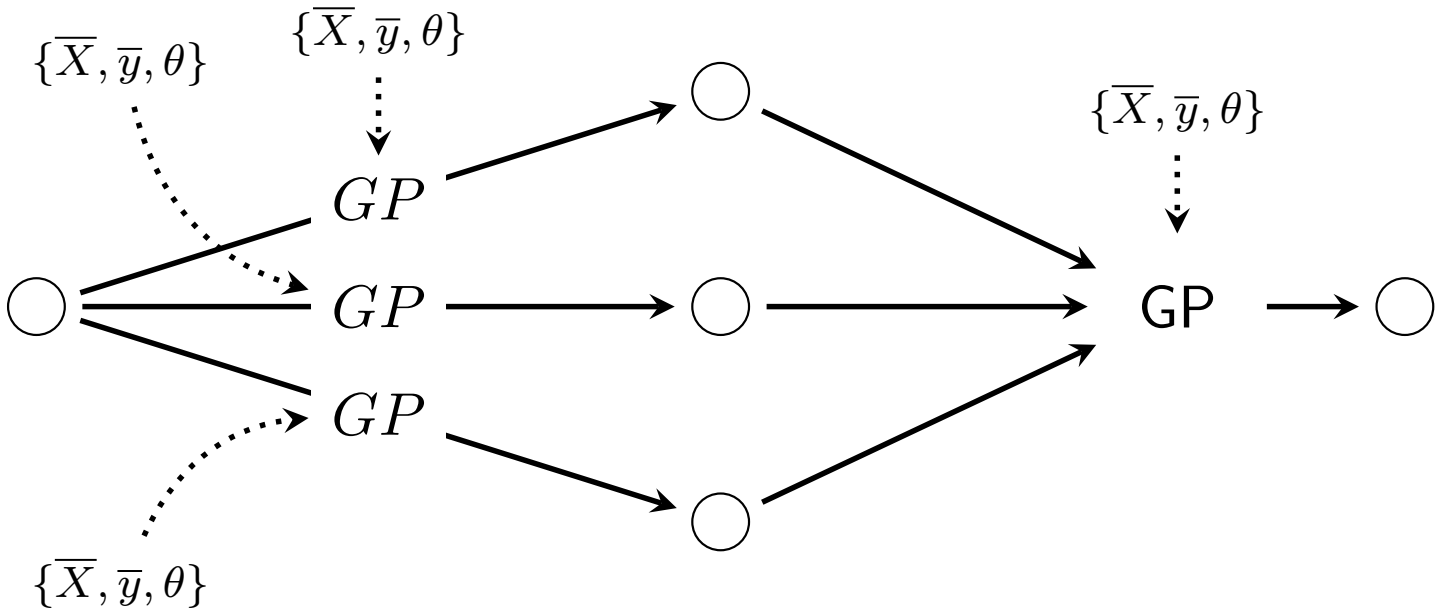
- Deep architectures compute layer-wise representations
- Single kernel parameters θ often insufficient



- Seperate inducing points and kernel parameters $\{\overline{X}, \overline{y}, \theta\}$

Excursion into Deep Gaussian Processes

- Deep architectures compute layer-wise representations
- Single kernel parameters θ often insufficient



- Seperate inducing points and kernel parameters $\{\overline{X}, \overline{y}, \theta\}$
- Sample hidden representation from posterior distribution