Fast Algorithms for Gaussian Processes and Bayesian Optimization

Leo Huang

21 October 2019

Outline

Bayesian Optimization

- Black box optimization method for expensive functions
- Surrogate model $\mathcal{GP}(\mu, K)$ for the objective
- Acquisition function $\mathcal{A}(\cdot)$ for choose next sampling point
- **Pros**: efficiently utilizes prior information
- **Cons**: cost of maintaining surrogate

Bayesian Optimization

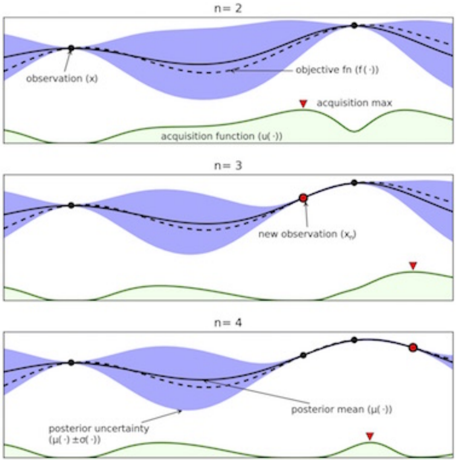


Figure: Taking the Human Out of the Loop: A Review of Bayesian Optimization (Shahriari et al.)

Gaussian Processes

- Finite collection of jointly normal random variables
- $f(x) \sim \mathcal{GP}(\mu, k)$ means that for any finite collection of function values $\mathbf{f} = f(\mathbf{X})$,

$$\mathbf{f} = [f(\mathbf{x}_1), ..., f(\mathbf{x}_n)] \sim \mathcal{N}(\mu, K)$$

- $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$

Gaussian Processes

- Associated mean function $\mu(x)$ and covariance/kernel function $k(x, x')$
- **Covariance/Kernel matrix** $K(\mathbf{x}, \mathbf{x}, \theta) =$

$$\begin{bmatrix} k(x_1, x_1, \theta) & k(x_1, x_2, \theta) & \dots & \dots & k(x_1, x_n, \theta) \\ k(x_2, x_1, \theta) & k(x_2, x_2, \theta) & & & \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ & & & & k(x_{n-1}, x_n, \theta) \\ k(x_n, x_1, \theta) & \dots & \dots & k(x_n, x_{n-1}, \theta) & k(x_n, x_n, \theta) \end{bmatrix}$$

Kernel Functions

- Gaussian/RBF/Squared Exponential

$$k(x, y, \ell, \sigma_{noise}, \sigma_{sf}) = \sigma_{sf}^2 \exp\left(-\frac{\|x-y\|^2}{\ell^2}\right) + \sigma_{noise}^2 I$$

- Matern

$$k(x, y, \sigma, \nu, \rho) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\frac{\|x-y\|}{\rho}\right)^{\nu} K_{\nu}\left(\sqrt{2\nu}\frac{\|x-y\|}{\rho}\right)$$

Gaussian Process Prediction

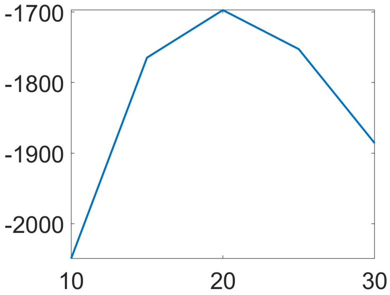
- $\begin{bmatrix} f \\ f^* \end{bmatrix} \sim \mathcal{N}\left(\mu, \begin{bmatrix} K + \sigma^2 I_n & K_* \\ K_*^T & K_{**} \end{bmatrix}\right)$
- Conditional Predictive Distribution in the Noisy Setting
- $\mathrm{Cov}(f^*) = K_{**} - K_*^T (K + \sigma^2 I_n)^{-1} K_*$
- $\mu(f^*) = K_*^T (K + \sigma^2 I_n)^{-1} f$

Marginal Likelihood

- Useful metric for evaluating evidence at a given point

$$
\begin{aligned}
p(\theta|y) &= \int p(\mathbf{y}|\mathbf{f}, X)p(\mathbf{f}|X)\,d\mathbf{f} \\
&= -\frac{1}{2}\mathbf{y}^T(K_\theta + \sigma_n^2 I)^{-1}\mathbf{y} - \frac{1}{2}\log|K_\theta + \sigma_n^2 I| - \frac{n}{2}\log 2\pi
\end{aligned}
$$

- Compute $\log\det(K)$ and $K^{-1}y$



Computing the Log Determinant

- $\log\det(A) = tr(\log A) = \sum \log(\lambda_i) = \mathbb{E}(z_i^T \log A z_i)$
- Approximation scheme for $\mathbb{E}(z_i^T \log A z_i)$
  - Use FFT, FMM in low dimensional space
- $\mathrm{tr}(f(A)) \approx \frac{1}{m}\sum_{i=1}^{m} v_i^T f(A) v_i$

Stochastic Trace Estimation

Theorem 0.1: Avron and Toledo

If v is a vector whose entries follow a Rademacher distribution, then

$$\mathbb{E}[v^T A v] = \mathrm{tr}(A)$$
$$\mathrm{Var}(v^T A v) = 2\sum_{i \neq j} A_{ij}^2 = 2(\|A\|_F - \sum_i A_{ii}^2)$$

Hutchinson's Estimator

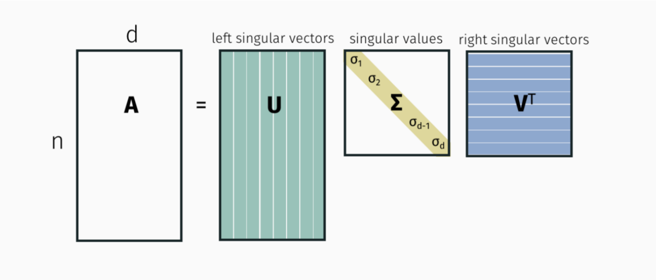
$$\log\det(A) = tr(\log A) = \sum \log(\lambda_i) = \mathbb{E}(z_i^T \log A z_i)$$

- Choice of A: Symmetric Positive Definite
- Choice of z:
  - z has independent random entries
  - z has zero mean and unit variance
  - Standard choices for the probe vector z:
    - Rademacher: $z_i = \pm 1$ with probability 0.5
    - Gaussian: $z_i \sim N(0, 1)$

Matrix Function

What is a matrix function:

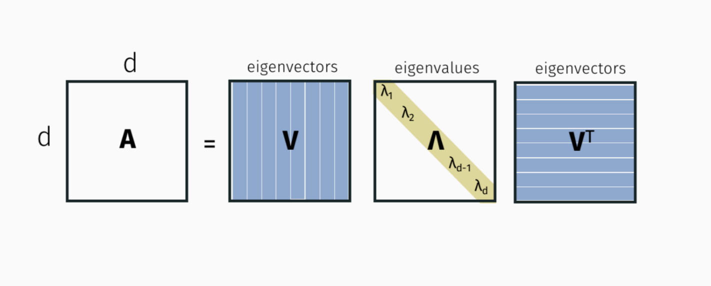
- Every matrix $A \in \mathbb{R}^{n \times d}$ has a singular value decomposition



- U,V are orthogonal, Σ is diagonal,$\sigma_1 \geq \ldots \geq \sigma_d \in \mathbb{R}^+$

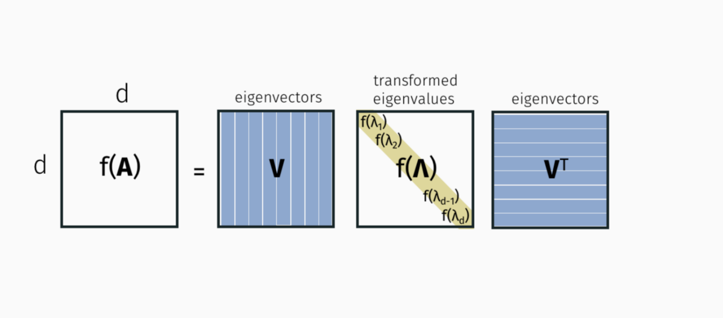
Matrix Function

- Every symmetric matrix $A \in \mathbb{R}^{d \times d}$ has an orthogonal eigendecomposition:



Matrix Function

- For any <u>scalar</u> function $f : \mathbb{R} \to \mathbb{R}$ define $f(A)$:



- $A = V\Lambda V^T \to f(A) = Vf(\Lambda)V^T$

Matrix Function

- Use Lanczos Method to approximate any matrix function
- Example: Least square regression
  $\underset{w}{\operatorname{argmin}} \sum |b_i - a_i^T w|^2 = ||Aw - b||^2$
  where $w = (A^T A)^{-1} A^T b = f(A^T A) A^T b$ where $f(t) = \frac{1}{t}$

Computing Matrix Function

Cost to compute f(A)

- eigendecomposition $A = V\Lambda V^T$: $O(n^3)$
- compute $f(\Lambda)$: $O(n)$
- form $f(A) = Vf(\Lambda)V^T$: $O(n^3)$

$O(n^3) + O(n) + O(n^3) = O(n^3)$ in practice

Lanczos Method for Matrix Functions

- Typically only interested in computing $f(A)x$ for some x$\in \mathbb{R}^n$
- Often much cheaper than computing f(A) explicitly

Approximate $f(A)$ using Lanczos method

- Fast: Three-term recurrence
- Reduce the problem to the cost of computing a matrix function for a $k \times k$ matrix.

Lanczos Method for Matrix Functions

- Step 1: Form orthogonal matrix $\mathrm{Q} = [q_0, q_1, ..., q_k]$ that spans the *Krylov* subspace
  - $K = \{x, Ax, A^2x, ..., A^kx\}$

Lanczos Method for Matrix Functions

- Step 1: Form orthogonal matrix Q $= [q_0, q_1, ..., q_k]$ that spans the *Krylov* subspace
  - $K = \{x, Ax, A^2x, ..., A^kx\}$
- Step 2: Compute $T = Q^TAQ$
  - Q has orthonormal columns
  - T is tridiagonal matrix

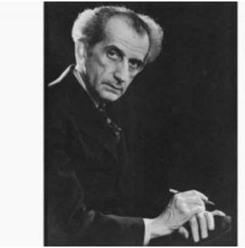
Lanczos Method for Matrix Functions

- Step 1: Form orthogonal matrix $Q = [q_0, q_1, ..., q_k]$ that spans the *Krylov* subspace
  - $K = \{x, Ax, A^2x, ..., A^kx\}$
- Step 2: Compute $T = Q^TAQ$
  - Q has orthonormal columns
  - T is tridiagonal matrix
- Step 3: Approximate $f(A)x$ by $Qf(T)Q^Tx$

Lanczos Algorithm

- Top Ten Algorithms of the 20th Century

- Metropolis Algorithm for Monte Carlo
- Simplex Method for Linear Programming
- Krylov Subspace Iteration Methods
- The Decompositional Approach to Matrix Computations
- The Fortran Optimizing Compiler
- QR Algorithm for Computing Eigenvalues
- Quicksort Algorithm for Sorting
- Fast Fourier Transform
- Integer Relation Detection
- Fast Multipole Method



Lanczos Algorithm

- Orthogonalize the Krylov matrix

$$\mathcal{K}(A, q_1, k) = \begin{bmatrix} q_1 & Aq_1 & \dots & A^{k-1}q_1 \end{bmatrix}$$

- Form Q_k, where the first j columns of Q_k spans $\mathcal{K}(A, q_1, j)$
- $q_1 = q_1/\|q_1\|$ and $Aq_1 = h_{21}q_2 + h_{11}q_1$

$$h_{21} = \|Aq_1 - h_{11}q_1\|$$

$$h_{j+1,j}q_{j+1} = Aq_j - \sum_{i=1}^{j} h_{ij}q_i$$

$$h_{ij} = \langle Aq_j, q_i \rangle, 1 \leq i \leq j$$

$$h_{j+1} = \|Aq_j - \sum_{i=1}^{j} h_{ij}q_j\|$$

Lanczos Algorithm

- Store h_{ij}s in an upper Hessenberg matrix $H_{k+1,k}$

$$H_{k+1,k} = \begin{bmatrix} h_{11} & h_{12} & \dots & \dots & h_{1k} \\ h_{21} & h_{22} & h_{32} & \dots & h_{2k} \\ 0 & h_{32} & h_{33} & \dots & h_{3k} \\ 0 & 0 & h_{43} & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & h_{k,k} \\ 0 & 0 & \dots & 0 & h_{k+1,k} \end{bmatrix}$$

- $AQ_k = Q_{k+1}H_{k+1,k} \;\; Q_k^T AQ_k = H_{k,k}$
- If A is symmetric, then $H_{k,k}$ is symmetric, and hence tridiagonal

Lanczos Algorithm

- $AQ_k = Q_{k+1} T_{k+1,k}$

$$T_{k+1,k} = \begin{bmatrix} \gamma_1 & \beta_1 & 0 & \dots & 0 \\ \beta_1 & \gamma_2 & \beta_2 & \dots & 0 \\ 0 & \beta_2 & \gamma_3 & \dots & \vdots \\ \vdots & 0 & \ddots & \ddots & \dots \\ 0 & \vdots & \ddots & \beta_{k-1} & \gamma_k \\ 0 & \dots & \dots & \dots & \beta_k \end{bmatrix}$$

- Three-term recurrence

$$\beta_j q_{j+1} = Aq_j - \gamma_j q_j - \beta_{j-1} q_{j-1}$$

Lanczos Algorithm

Theorem 0.2: Lanczos Polynomial

v_k, the kth Lanczos vector, is a polynomial p_k in A applied to v_1.

$$\begin{aligned} v_k &= p_k(A)v_1 \\ p_k(\lambda) &= (-1)^{k-1}\frac{\det(T_{k-1} - \lambda I)}{\beta_1, \ldots \beta_{k-1}}, k > 1, \\ p_0(\lambda) &= 1 \end{aligned}$$

Lanczos Algorithm

Theorem 0.3: The Lanczos Algorithm Implicitly Defines a Measure

Let v_k and v_l be Lanczos vectors. Then there exists a measure α such that

$$\langle v_k, v_l \rangle = \langle p_k, p_l \rangle = \int_a^b p_k(\lambda) p_l(\lambda)\, d\alpha(\lambda)$$

where $a \leq \lambda_1 = \lambda_{min}$ and $\lambda_{max} = \lambda_n \leq b$ and p_i is the polynomial associated with the ith Lanczos vector.

Lanczos Algorithm

Proof.

- Spectral decomposition $A = Q\Lambda Q^T$
- $P_k(A) = QP_k(\lambda)Q^T$.

$$\begin{aligned}
\langle v_k, v_l \rangle &= (v_1)^T P_k(A)^T P_l(A) v_1 \\
&= (v_1)^T Q P_k(\Lambda) Q^T Q P_l(\Lambda) Q^T v_1 \\
&= (v_1)^T Q P_k(\Lambda) P_l(\Lambda) Q^T v_1 \\
&= \sum_{j=1}^{n} P_k(\lambda_j) P_l(\lambda_j) (Q^T v_1)_j^2
\end{aligned}$$

Hence we may define

$$\alpha(\lambda) = \begin{cases} 0 & \lambda < \lambda_1 \\ \sum_{j=1}^{i} (Q^T v_1)_j^2 & \lambda_i \leq \lambda < \lambda_{i+1} \\ \sum_{j=1}^{n} (Q^T v)_j^2 & \lambda_n \leq \lambda \end{cases}$$

Linear System Solve

$$p(\theta|y) = -\frac{1}{2}\mathbf{y}^T(K_\theta + \sigma_n^2 I)^{-1}\mathbf{y} - \frac{1}{2}\log|K_\theta + \sigma_n^2 I| - \frac{n}{2}\log 2\pi$$

- $(K^{-1}y)$
- Preconditioned Conjugate Gradients (PCG)
- $Ax = b \implies PAx = Pb$
- $K = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & \\ & \Sigma_2 \end{bmatrix} \begin{bmatrix} U_1 & U_2 \end{bmatrix}^T$
- $P = U_1 \Sigma_1^T U_1^T + \sigma^2 I$

$$P^{-1}K = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} I_k & \\ & \sigma^{-1}(\Sigma_2 + \sigma I_{n-k}) \end{bmatrix} \begin{bmatrix} U_1 & U_2 \end{bmatrix}^T$$

Bayesian Optimization Acquisition Function

- Expected Improvement

$$\begin{aligned}
\mathrm{EI}_n(x) &= \mathbb{E}_n[\max(0, f - f_n^*)] \\
&= \int_{f_n^*}^{\infty} (f - f^*) \left\{ \frac{1}{\sigma} \varphi \left(\frac{f - \mu}{\sigma} \right) \right\} df \\
&= \sigma(Z + Z\Phi(Z))
\end{aligned}$$

  - $Z = \frac{\mu - f}{\sigma}$ and φ and Φ are the standard normal PDF and CDF
- Lower/Upper Confidence Bound

$$f(x) = \mu(x) + \kappa \mathrm{cov}(x)$$

  - κ is exploration/exploitation tradeoff hyperparameter

References

- *Yousef Saad*, Approximating Spectral Densities of Large Matrices
- *Christopher Musco*, The Lanczos Method in Data Science
- *Yuanzhe Xi*, Fast computation of spectral densities for generalized eigenvalue problems
- *Gene Golub and Charles Van Loan*, Matrix Computations
- *David Bindel*, Stochastic Linear Algebra for Scalable Gaussian Processes
- *Haim Avron and Sivian Toledo*, Estimating the Trace of an Implicit Matrix
- *Uri Ascher and Chen Greif*, A First Course in Numerical Methods