Sampling from Gaussian Process Posteriors using Stochastic Gradient Descent
Gaussian Processes

Probabilistic formulation provides uncertainty
Bayesian Optimization

Automatic explore-exploit tradeoff
From Bayesian Optimization to Bayesian Interactive Decision-making
Pathwise Conditioning

\[ (f \mid y)(\cdot) = f(\cdot) + \sum_{i=1}^{N} v_i k(x_i, \cdot) \]

\[ v = K_{xx}^{-1}(y - f(x)) \]

\( v \): representer weights \hspace{1cm} k(x_i, \cdot): canonical basis functions
Conjugate Gradients

Refinement of gradient descent for solving linear systems $\mathbf{A}^{-1}\mathbf{b}$
Convergence rate is much faster than gradient descent
Precise rate depends mainly on $\text{cond}(\mathbf{A})$
Numerical Stability

Condition number: quantifies difficulty of solving $A^{-1}b$

$$\text{cond}(A) = \lim_{\varepsilon \to 0} \sup_{\|\delta\| \leq \varepsilon \|b\|} \frac{\|A^{-1}(b + \delta) - A^{-1}b\|_2}{\varepsilon \|A^{-1}b\|_2} = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$$

$\lambda_{\min}, \lambda_{\max}$: eigenvalues
Condition Numbers of Kernel Matrices

Are kernel matrices always well-conditioned? **No.**

One-dimensional time series on grid: Kac–Murdock–Szegö matrix

\[
K_{xx} = \begin{pmatrix}
1 & \rho & \rho^2 & \ldots & \rho^{n-1} \\
\rho & 1 & \rho & \ldots & \rho^{n-2} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\rho^{n-1} & \rho^{n-2} & \rho^{n-3} & \ldots & 1
\end{pmatrix}
\]

for which

\[
\frac{1+2\rho+2\varepsilon+\rho^2}{1-2\rho-2\varepsilon+\rho^2} \leq \text{cond}(K_{xx}) \leq \frac{(1+\rho)^2}{(1-\rho)^2},
\]

where \(\varepsilon = \frac{\pi^2}{N+1}\).
Condition Numbers of Kernel Matrices

Problem: too much correlation $\rightsquigarrow$ points too close by
Minimum Separation

*Separation:* minimum distance between distinct $z_i$ and $z_j$

**Proposition.** Assuming spatial decay and stationarity, the condition controls $\text{cond}(\mathbf{K}_{zz})$ uniformly in $\mathcal{M}$.

Idea: use this to select numerically stable inducing points
Sampling from Gaussian Process Posteriors using Stochastic Gradient Descent

Jihao Andreas Lin,* Javier Antorán,* Shreyas Padhy,* David Janz, José Miguel Hernández-Lobato, Alexander Terenin

*equal contribution
Have you tried stochastic gradient descent?

Conventional wisdom in deep learning:

- SGD variants are empirically often the best optimization algorithms
- ADAM is extremely effective, even on non-convex problems
- Minibatch-based training critical part of scalability

Why not try it out for Gaussian process posterior sample paths?
Gaussian Process Posteriors via Randomized Optimization Objectives

Split into posterior mean and uncertainty reduction terms

\[
(f | y)(\cdot) = f(\cdot) + K_{(\cdot) x}(K_{xx} + \Sigma)^{-1}(y - f(x) - \varepsilon)
\]

\[
= f(\cdot) + \sum_{i=1}^{N} v_i^* k(x_i, \cdot) + \sum_{i=1}^{N} \alpha_i^* k(x_i, \cdot)
\]

where

\[
v^* = \arg \min_{v \in \mathbb{R}^N} \sum_{i=1}^{N} \frac{(y_i - K_{x_i x} v)^2}{\Sigma_{ii}} + v^T K_{xx} v
\]

\[
\alpha^* = \arg \min_{\alpha \in \mathbb{R}^N} \sum_{i=1}^{N} \frac{(f(x_i) + \varepsilon_i - K_{x_i x} \alpha)^2}{\Sigma_{ii}} + \alpha^T K_{xx} \alpha.
\]
Gaussian Process Posterior via Randomized Optimization Objectives

\[
\begin{align*}
\boldsymbol{v}^* &= \arg \min_{\boldsymbol{v} \in \mathbb{R}^N} \sum_{i=1}^{N} \frac{(y_i - \mathbf{K}_{x_i \alpha} \boldsymbol{v})^2}{\Sigma_{ii}} + \boldsymbol{v}^T \mathbf{K}_{xx} \boldsymbol{v} \\
\boldsymbol{\alpha}^* &= \arg \min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \sum_{i=1}^{N} \frac{(f(x_i) + \varepsilon_i - \mathbf{K}_{x_i \alpha} \boldsymbol{\alpha})^2}{\Sigma_{ii}} + \boldsymbol{\alpha}^T \mathbf{K}_{xx} \boldsymbol{\alpha}.
\end{align*}
\]

First term: apply mini-batch estimation
Second term: evaluate stochastically via random Fourier features
Variance reduction trick: shift \(\varepsilon_i\) into regularizer
Use stochastic gradient descent with Polyak averaging
Stochastic Gradient Descent

It works better than conjugate gradients on test data? *Wait, what?*
What happens in one dimension?

Performance depends on data-generation asymptotics
What happens in one dimension?

SGD does not converge to the correct solution, but still produces reasonable error bars...
Convergence: Euclidean and RKHS Norms

No convergence in representer weight space or in the reproducing kernel Hilbert space

Good test performance
Convergence: Euclidean and RKHS Norms

Performance not significantly affected by noise
Unstable optimization problem $\leadsto$ benign non-convergence $\leadsto$ implicit bias
Where does SGD's implicit bias affect predictions?

Error seems to concentrate away from data, but not too far away?
Where does SGD's implicit bias affect predictions?

Interpolation region  Extrapolation region  Far-away region
The Far-away Region

\[(f \mid y)(\cdot) = f(\cdot) + \sum_{i=1}^{N} v_i k(x_i, \cdot)\]

Kernel decays in space \(\sim\) predictions revert to prior
The Interpolation Region

Low approximation error where data is dense
The Interpolation Region

Idea: maybe SGD (a) converges fast on a subspace, and (b) obtains something *arbitrary but benign* on the rest of the space?

Let $K_{xx} = U \Lambda U^T$ be the eigendecomposition of the kernel matrix. Define the *spectral basis functions*

$$u^{(i)}(\cdot) = \sum_{j=1}^{N} \frac{U_{ji}}{\sqrt{\lambda_i}} k(x_j, \cdot).$$
Large-eigenvalue spectral basis functions concentrate on data
The Interpolation Region

**Proposition.** With probability $1 - \delta$, we have

$$\left\| \text{proj}_{u^{(i)}} \mu_{f|y} - \text{proj}_{u^{(i)}} \mu_{\text{SGD}} \right\|_{H_k} \leq \frac{1}{\sqrt{\lambda_i} t} \left( \frac{\| y \|_2}{\eta \sigma^2} + G \sqrt{2\eta \sigma^2 \log \frac{N}{\delta}} \right).$$

η: learning rate  \quad σ^2: noise variance  \quad \lambda_i: kernel matrix eigenvalues  \quad G: gradient's sub-Gaussian coefficient

SGD converges fast with respect to top spectral basis functions
The Interpolation Region

Where do the top spectral basis functions concentrate?

- Idea: lift Courant–Fischer eigenvector characterization to the RKHS

**Proposition.** The spectral basis functions can be written

\[
    u^{(i)}(\cdot) = \arg \max_{u \in H_k} \left\{ \sum_{i=1}^{N} u(x_i)^2 : \|u\|_{H_k} = 1, \langle u, u^{(j)} \rangle_{H_k} = 0, \forall j < i \right\}.
\]

Spectral basis functions concentrate on the data as much as possible, while remaining orthogonal to those with larger eigenvalues
The Extrapolation Region

High error: where small-eigenvalue spectral basis functions concentrate
SGD's implicit bias for Gaussian processes

Implicit bias: SGD converges quickly near the data, and causes no harm far from the data

Error concentrates in regions (a) without much data, which also (b) aren't located too far from the data:

- Lack of data $\implies$ predictions are mostly arbitrary
- Empirically: functions shrink to prior faster than exact posterior

Benin non-convergence $\implies$ robustness to instability
Performance

Conjugate gradients: non-monotonic test error, in spite of monotonic convergence
SGD: almost always monotonic decrease in test error
## Performance

<table>
<thead>
<tr>
<th>Dataset</th>
<th>POL</th>
<th>ELEVATORS</th>
<th>BIKE</th>
<th>PROTEIN</th>
<th>KEGGDIR</th>
<th>3DROAD</th>
<th>SONG</th>
<th>BUZZ</th>
<th>HOUSEELEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>15000</td>
<td>16599</td>
<td>17379</td>
<td>45730</td>
<td>48827</td>
<td>434874</td>
<td>515345</td>
<td>583250</td>
<td>2049280</td>
</tr>
<tr>
<td><strong>RMSE</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SGD</td>
<td>0.13±0.00</td>
<td>0.38±0.00</td>
<td>0.11±0.00</td>
<td><strong>0.51±0.00</strong></td>
<td>0.12±0.00</td>
<td><strong>0.11±0.00</strong></td>
<td>0.80±0.00</td>
<td>0.42±0.01</td>
<td><strong>0.09±0.00</strong></td>
</tr>
<tr>
<td>CG</td>
<td><strong>0.08±0.00</strong></td>
<td><strong>0.35±0.00</strong></td>
<td><strong>0.04±0.00</strong></td>
<td><strong>0.50±0.00</strong></td>
<td><strong>0.08±0.00</strong></td>
<td>0.15±0.01</td>
<td>0.85±0.03</td>
<td>1.41±0.08</td>
<td>0.87±0.14</td>
</tr>
<tr>
<td>SVGP</td>
<td>0.10±0.00</td>
<td>0.37±0.00</td>
<td>0.08±0.00</td>
<td>0.62±0.00</td>
<td>0.10±0.00</td>
<td>0.64±0.01</td>
<td>0.82±0.00</td>
<td><strong>0.34±0.00</strong></td>
<td>0.10±0.02</td>
</tr>
<tr>
<td><strong>RMSE+$\dagger$</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SGD</td>
<td><strong>0.13±0.00</strong></td>
<td><strong>0.38±0.00</strong></td>
<td>0.11±0.00</td>
<td><strong>0.51±0.00</strong></td>
<td><strong>0.12±0.00</strong></td>
<td><strong>0.11±0.00</strong></td>
<td>0.80±0.00</td>
<td><strong>0.42±0.01</strong></td>
<td><strong>0.09±0.00</strong></td>
</tr>
<tr>
<td>CG</td>
<td>0.16±0.01</td>
<td>0.68±0.09</td>
<td><strong>0.05±0.01</strong></td>
<td>3.03±0.23</td>
<td>9.79±1.06</td>
<td>0.34±0.02</td>
<td>0.83±0.02</td>
<td>5.66±1.14</td>
<td>0.93±0.19</td>
</tr>
<tr>
<td>SVGP</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td><strong>Hours</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SGD</td>
<td>0.06±0.00</td>
<td>0.06±0.00</td>
<td>0.09±0.00</td>
<td>0.12±0.00</td>
<td>0.25±0.00</td>
<td>0.46±0.19</td>
<td>3.67±0.24</td>
<td>5.78±1.02</td>
<td>2.69±0.91</td>
</tr>
<tr>
<td>CG</td>
<td>0.04±0.01</td>
<td><strong>0.03±0.01</strong></td>
<td>0.05±0.00</td>
<td>0.15±0.03</td>
<td>0.21±0.03</td>
<td>1.42±0.60</td>
<td>3.25±0.04</td>
<td>5.85±0.80</td>
<td>2.62±0.01</td>
</tr>
<tr>
<td>SVGP</td>
<td><strong>0.04±0.00</strong></td>
<td><strong>0.04±0.00</strong></td>
<td><strong>0.04±0.00</strong></td>
<td><strong>0.04±0.00</strong></td>
<td><strong>0.04±0.00</strong></td>
<td><strong>0.04±0.00</strong></td>
<td><strong>0.05±0.00</strong></td>
<td><strong>0.04±0.00</strong></td>
<td><strong>0.04±0.00</strong></td>
</tr>
<tr>
<td><strong>NLL</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SGD</td>
<td>-0.70±0.02</td>
<td>0.47±0.00</td>
<td>-0.48±0.08</td>
<td>0.64±0.01</td>
<td>-0.62±0.07</td>
<td><strong>-0.60±0.00</strong></td>
<td><strong>1.21±0.00</strong></td>
<td>0.83±0.07</td>
<td><strong>-1.09±0.04</strong></td>
</tr>
<tr>
<td>CG</td>
<td><strong>-1.17±0.01</strong></td>
<td><strong>0.38±0.00</strong></td>
<td><strong>-2.62±0.06</strong></td>
<td><strong>0.62±0.01</strong></td>
<td><strong>-0.92±0.10</strong></td>
<td>16.27±0.45</td>
<td>1.36±0.07</td>
<td>2.38±0.08</td>
<td>2.07±0.58</td>
</tr>
<tr>
<td>SVGP</td>
<td>-0.64±0.02</td>
<td>0.44±0.00</td>
<td>-1.47±0.02</td>
<td>0.94±0.01</td>
<td><strong>-0.89±0.03</strong></td>
<td>0.94±0.03</td>
<td>1.23±0.00</td>
<td><strong>0.29±0.04</strong></td>
<td>-0.94±0.13</td>
</tr>
</tbody>
</table>

Strong predictive performance at sufficient scale
Parallel Thompson Sampling

Uncertainty: strong decision-making performance
Reflections

Numerical analysis conventional wisdom:
- Don't run gradient descent on quadratic objectives
  - It's slow, conjugate gradient works much better
- If CG is slow then your problem is unstable
  - Unstable problems are ill-posed, you should reformulate

This work: a very different way of looking at things
- Don't solve the linear system approximately if the solution isn't inherently needed
- Instead of a well-posed problem, a well-posed subproblem might be good enough
- Try SGD! It might work well, including for counterintuitive reasons
- A kernel matrix's eigenvectors carry information about data-density
  - To see this, adopt a function-analytic view given by the spectral basis functions
Thank you!

HTTPS://AVT.IM/ · @AVT_IM


*Equal contribution