Advances in kernel exponential families

Arthur Gretton

Gatsby Computational Neuroscience Unit,
University College London

Tenerife, 2018
Motivating application:

- Fast estimation of complex multivariate densities

The infinite exponential family:

- Multivariate Gaussian → Gaussian process
- Finite mixture model → Dirichlet process mixture model
- Finite exponential family → ???

Application:

- Adaptive HMC for Pseudo-Marginal MCMC (likelihood not computable), or amortized HMC

In this talk:

- Fitting of the infinite dimensional exponential family using score matching
  Sriperumbudur, Fukumizu, G., Hyvarinen, Kumar, JMLR (2017)
- Guaranteed speed improvements by Nystrom
  Sutherland, Hyvarinen, Arbel, G., AISTATS (2018)
- Conditional models
  Arbel, G., AISTATS (2018)
- Deep infinite exponential family
  Li, Sutherland, Strathmann, G., ??? (2023)
Outline

Motivating application:
- Fast estimation of complex multivariate densities

The infinite exponential family:
- Multivariate Gaussian → Gaussian process
- Finite mixture model → Dirichlet process mixture model
- Finite exponential family → ???

Application:
- Adaptive HMC for Pseudo-Marginal MCMC (likelihood not computable), or amortized HMC

In this talk:
- Fitting of the infinite dimensional exponential family using score matching Sriperumbudur, Fukumizu, G, Hyvarinen, Kumar, JMLR (2017)
- Guaranteed speed improvements by Nystrom Sutherland, Hyvarinen, Arbel, G., AISTATS (2018)
- Conditional models Arbel, G., AISTATS (2018)
- Deep infinite exponential family Li, Sutherland, Strathmann, G., ??? (2023)
Goal 1: learn high dimensional, complex densities

We want:

- Efficient computation and representation
- Statistical guarantees
Goal 2: adaptive hamiltonian monte carlo

- HMC: distant moves, high acceptance probability.

- Potential energy
  \[ U(x) = - \log \pi(x) \], auxiliary momentum \( p \sim \exp(-K(p)) \),
  simulate for \( t \in \mathbb{R} \) along Hamiltonian flow of
  \[ H(p, x) = K(p) + U(x) \],
  using operator
  \[
  \frac{\partial K}{\partial p} \frac{\partial}{\partial x} - \frac{\partial U}{\partial x} \frac{\partial}{\partial p}
  \]

- Numerical simulation (i.e. leapfrog) depends on gradient information.
Goal 2: adaptive hamiltonian monte carlo

Sliced posterior over hyperparameters of a Gaussian Process classifier on UCI Glass dataset obtained using Pseudo-Marginal MCMC.

Can you learn an HMC sampler?
The exponential family

The exponential family in $\mathbb{R}^d$

$$p(x) = \exp \left( \left\langle \begin{array}{c} \eta \\ \text{natural parameter} \\ \text{sufficient statistic} \\ \text{log normaliser} \\ \text{base measure} \end{array} \right\rangle - A(\eta) \right) \frac{1}{q_0(x)}$$

Examples:
- Gaussian density: $T(x) = \begin{bmatrix} x & x^2 \end{bmatrix}$
- Gamma density: $T(x) = \begin{bmatrix} \ln x & x \end{bmatrix}$

Can we extend this to infinite dimensions?
The kernel exponential family


\[ \mathcal{P} = \left\{ p_f(x) = e^{\langle f, \varphi(x) \rangle \mathcal{H} - A(f)} q_0(x), \ x \in \Omega, f \in \mathcal{F} \right\} \]

where

\[ \mathcal{F} = \left\{ f \in \mathcal{H} : A(f) = \log \int e^{f(x)} q_0(x) \, dx < \infty \right\} \]
The kernel exponential family


\[
P = \left\{ p_f(x) = e^{\langle f, \varphi(x) \rangle_H - A(f)} q_0(x), \ x \in \Omega, f \in \mathcal{F} \right\}
\]

where

\[
\mathcal{F} = \left\{ f \in H : A(f) = \log \int e^{f(x)} q_0(x) \, dx < \infty \right\}
\]

Finite dimensional RKHS: one-to-one correspondence between finite dimensional exponential family and RKHS.

Example: Gaussian kernel, \( T(x) = \begin{bmatrix} x & x^2 \end{bmatrix} = \varphi(x) \) and \( k(x, y) = xy + x^2y^2 \)
Given random samples, $X_1, \ldots, X_n$ drawn i.i.d. from an unknown density, $p_0 := p_{f_0} \in \mathcal{P}$, estimate $p_0$
How not to do it: maximum likelihood

Maximum likelihood:

\[
f_{ML} = \arg \max_{f \in \mathcal{F}} \sum_{i=1}^{n} \log p_f(X_i)
\]

\[
= \arg \max_{f \in \mathcal{F}} \sum_{i=1}^{n} f(X_i) - n \log \int e^{f(x)} q_0(x) \, dx.
\]
How not to do it: maximum likelihood

Maximum likelihood:

\[
f_{ML} = \arg \max_{f \in \mathcal{F}} \sum_{i=1}^{n} \log p_f(X_i)
\]

\[
= \arg \max_{f \in \mathcal{F}} \sum_{i=1}^{n} f(X_i) - n \log \int e^{f(x)} q_0(x) \, dx.
\]

Solving the above yields that \( f_{ML} \) satisfies

\[
\frac{1}{n} \sum_{i=1}^{n} \phi(x_i) = \int \phi(x) p_{f_{ML}}(x) \, dx
\]

where \( p_{f_{ML}} = \frac{dp_{ML}}{dx} \).

Ill posed for infinite dimensional \( \phi(x) \)!
Loss is **Fisher Score**:

\[
D_F(p_0, p_f) := \frac{1}{2} \int_\Omega p_0(x) \left\| \nabla_x \log p_0(x) - \nabla_x \log p_f(x) \right\|^2 \, dx
\]

**Domain** is $\Omega$: open subset of $\mathbb{R}^d$ with piecewise smooth boundary $\partial \Omega := \overline{\Omega} \setminus \Omega$, 
Score matching (general version)

Assuming $p_f$ to be twice differentiable (w.r.t. $x$) and
\[ \int p_0(x) ||\nabla x \log p_f(x)||^2 \, dx < \infty, \forall \theta \in \Theta \]

\[ D_F(p_0, p_f) := \frac{1}{2} \int p_0(x) ||\nabla x \log p_0(x) - \nabla x \log p_f(x)||^2 \, dx \]
\[ \overset{(a)}{=} \int p_0(x) \sum_{i=1}^{d} \left( \frac{1}{2} \left( \frac{\partial \log p_f(x)}{\partial x_i} \right)^2 + \frac{\partial^2 \log p_f(x)}{\partial x_i^2} \right) \, dx \]
\[ + \frac{1}{2} \int p_0(x) \left\| \frac{\partial \log p_0(x)}{\partial x} \right\|^2 \, dx \]

where partial integration is used in (a) under mild conditions:

1. $p_0$ continuously extendible to $\overline{\Omega}$.
2. kernel $k$ twice continuously differentiable on $\Omega \times \Omega$ with continuous extension of $\partial^\alpha \partial^\alpha k$ to $\overline{\Omega} \times \overline{\Omega}$ for $|\alpha| \leq 2$.
3. $\partial_i \partial_{i+d} k(x, x)p_0(x) = 0$ for $x \in \partial \Omega$ and $\sqrt{\partial_i \partial_{i+d} k(x, x)p_0(x)} = o(||x||_2^{1-d})$ as $x \in \Omega$, $||x||_2 \to \infty$, $\forall i \in [d]$. 

11/66
Score matching: 1-D proof

\[ D_F(p_0, p_f) \]
\[ = \frac{1}{2} \int_a^b p_0(x) \left( \frac{d \log p_0(x)}{dx} - \frac{d \log p_f(x)}{dx} \right)^2 dx \]
Score matching: 1-D proof

\[ D_F(p_0, p_f) \]

\[ = \frac{1}{2} \int_a^b p_0(x) \left( \frac{d \log p_0(x)}{dx} - \frac{d \log p_f(x)}{dx} \right)^2 dx \]

\[ = \frac{1}{2} \int_a^b p_0(x) \left( \frac{d \log p_0(x)}{dx} \right)^2 dx + \frac{1}{2} \int_a^b p_0(x) \left( \frac{d \log p_f(x)}{dx} \right)^2 dx \]

\[ - \int_a^b p_0(x) \left( \frac{d \log p_f(x)}{dx} \right) \left( \frac{d \log p_0(x)}{dx} \right) dx \]
Score matching: 1-D proof

\[ D_F(p_0, p_f) \]

\[ = \frac{1}{2} \int_a^b p_0(x) \left( \frac{d \log p_0(x)}{dx} - \frac{d \log p_f(x)}{dx} \right)^2 \, dx \]

\[ = \frac{1}{2} \int_a^b p_0(x) \left( \frac{d \log p_0(x)}{dx} \right)^2 \, dx + \frac{1}{2} \int_a^b p_0(x) \left( \frac{d \log p_f(x)}{dx} \right)^2 \, dx \]

\[ - \int_a^b p_0(x) \left( \frac{d \log p_f(x)}{dx} \right) \left( \frac{d \log p_0(x)}{dx} \right) \, dx \]

Final term:

\[ \int_a^b p_0(x) \left( \frac{d \log p_f(x)}{dx} \right) \left( \frac{d \log p_0(x)}{dx} \right) \, dx \]

\[ = \int_a^b p_0(x) \left( \frac{d \log p_f(x)}{dx} \right) \left( \frac{1}{p_0(x)} \frac{dp_0(x)}{dx} \right) \, dx \]

\[ = \left[ \left( \frac{d \log p_f(x)}{dx} \right) p_0(x) \right]_a^b - \int_a^b p_0(x) \frac{d^2 \log p_f(x)}{dx^2} \, dx \]
Score matching: 1-D proof

\[ D_F(p_0, p_f) \]

\[ = \frac{1}{2} \int_a^b p_0(x) \left( \frac{d \log p_0(x)}{dx} - \frac{d \log p_f(x)}{dx} \right)^2 \, dx \]

\[ = \frac{1}{2} \int_a^b p_0(x) \left( \frac{d \log p_0(x)}{dx} \right)^2 \, dx + \frac{1}{2} \int_a^b p_0(x) \left( \frac{d \log p_f(x)}{dx} \right)^2 \, dx \]

\[ - \int_a^b p_0(x) \left( \frac{d \log p_f(x)}{dx} \right) \left( \frac{d \log p_0(x)}{dx} \right) \, dx \]

Final term:

\[ \int_a^b p_0(x) \left( \frac{d \log p_f(x)}{dx} \right) \left( \frac{d \log p_0(x)}{dx} \right) \, dx \]

\[ = \int_a^b p_0(x) \left( \frac{d \log p_f(x)}{dx} \right) \left( \frac{1}{p_0(x)} \frac{dp_0(x)}{dx} \right) \, dx \]

\[ = \left[ \left( \frac{d \log p_f(x)}{dx} \right) p_0(x) \right]_a^b - \int_a^b p_0(x) \frac{d^2 \log p_f(x)}{dx^2} \, dx \]
Score matching: 1-D proof

\[ D_F(p_0, p_f) \]

\[ = \frac{1}{2} \int_a^b p_0(x) \left( \frac{d \log p_0(x)}{dx} - \frac{d \log p_f(x)}{dx} \right)^2 \, dx \]

\[ = \frac{1}{2} \int_a^b p_0(x) \left( \frac{d \log p_0(x)}{dx} \right)^2 \, dx + \frac{1}{2} \int_a^b p_0(x) \left( \frac{d \log p_f(x)}{dx} \right)^2 \, dx \]

\[ - \int_a^b p_0(x) \left( \frac{d \log p_f(x)}{dx} \right) \left( \frac{d \log p_0(x)}{dx} \right) \, dx \]

Final term:

\[ \int_a^b p_0(x) \left( \frac{d \log p_f(x)}{dx} \right) \left( \frac{d \log p_0(x)}{dx} \right) \, dx \]

\[ = \int_a^b p_0(x) \left( \frac{d \log p_f(x)}{dx} \right) \left( \frac{1}{p_0(x)} \frac{dp_0(x)}{dx} \right) \, dx \]

\[ = \left[ \left( \frac{d \log p_f(x)}{dx} \right) p_0(x) \right]_a^b - \int_a^b p_0(x) \frac{d^2 \log p_f(x)}{dx^2} \, dx \]
Relation to KL

Relation between Fisher score and KL:

**Proposition B.1** Let $p$ and $q$ be probability densities defined on $\mathbb{R}^d$. Define $p_t := p \ast N(0, tI_d)$ and $q_t := q \ast N(0, tI_d)$ where $N(0, tI_d)$ denotes a normal distribution on $\mathbb{R}^d$ with mean zero and diagonal covariance with $t > 0$. Suppose $p_t$ and $q_t$ satisfy

$$\partial_ip_t(x) \log p_t(x) = o(\|x\|_2^\alpha), \quad \partial_ip_t(x) \log q_t(x) = o(\|x\|_2^\alpha) \quad \text{and} \quad \partial_i \log q_t(x)p_t(x) = o(\|x\|_2^\alpha)$$

as $\|x\|_2 \to \infty$ for all $i \in [d]$ where $\alpha = 1 - d$. Then

$$KL(p\|q) = \int_0^\infty J(p_t\|q_t) \, dt, \quad (B.1)$$

where $J$ is defined in (3).

Sriperumbudur, Fukumizu, G, Hyvarinen, Kumar, JMLR (2017), but effectively from Lyu (2009)
Empirical score matching

$p_n$ represents $n$ i.i.d. samples from $P_0$

$$D_F(p_n, p_f) := \frac{1}{n} \sum_{a=1}^{n} \sum_{i=1}^{d} \left( \frac{1}{2} \left( \frac{\partial \log p_f(X_a)}{\partial x_i} \right)^2 + \frac{\partial^2 \log p_f(X_a)}{\partial x_i^2} \right) + C$$

Since $D_F(p_n, p_f)$ is independent of $A(f)$,

$$f_n^* = \arg \min_{f \in \mathcal{F}} D_F(p_n, p_f)$$

is well posed, unlike the MLE.
Empirical score matching

$p_n$ represents $n$ i.i.d. samples from $P_0$

$$D_F(p_n, p_f) := \frac{1}{n} \sum_{a=1}^{n} \sum_{i=1}^{d} \left( \frac{1}{2} \left( \frac{\partial \log p_f(X_a)}{\partial x_i} \right)^2 + \frac{\partial^2 \log p_f(X_a)}{\partial x_i^2} \right) + C$$

Since $D_F(p_n, p_f)$ is independent of $A(f)$,

$$f_n^* = \arg\min_{f \in \mathcal{F}} D_F(p_n, p_f)$$

is well posed, unlike the MLE.

Add extra term $\lambda \|f\|_2^2$ to regularize.
A kernel solution

Infinite exponential family:

\[ p_f(x) = e^{\langle f, \varphi(x) \rangle_{\mathcal{H}} - A(f)} q_0(x) \]

Thus

\[ \frac{\partial}{\partial x} \log p_f(x) = \frac{\partial}{\partial x} \langle f, \varphi(x) \rangle_{\mathcal{H}} + \frac{\partial}{\partial x} \log q_0(x). \]
A kernel solution

Infinite exponential family:

\[ p_f(x) = e^{\langle f, \varphi(x) \rangle_H} - A(f) q_0(x) \]

Thus

\[ \frac{\partial}{\partial x} \log p_f(x) = \frac{\partial}{\partial x} \langle f, \varphi(x) \rangle_H + \frac{\partial}{\partial x} \log q_0(x). \]

Kernel trick for derivatives: \[ \frac{\partial}{\partial x_i} f(X) = \langle f, \frac{\partial}{\partial x_i} \varphi(X) \rangle_H \]

Dot product between feature derivatives:

\[ \left\langle \frac{\partial}{\partial x_i} \varphi(X), \frac{\partial}{\partial x_j} \varphi(X') \right\rangle_H = \frac{\partial^2}{\partial x_i \partial x_{d+j}} k(X, X') \]
A kernel solution

Infinite exponential family:

\[ p_f(x) = e^{\langle f, \varphi(x) \rangle_H - A(f)} q_0(x) \]

Thus

\[ \frac{\partial}{\partial x} \log p_f(x) = \frac{\partial}{\partial x} \langle f, \varphi(x) \rangle_H + \frac{\partial}{\partial x} \log q_0(x). \]

Kernel trick for derivatives:

\[ \frac{\partial}{\partial x_i} f(X) = \langle f, \frac{\partial}{\partial x_i} \varphi(X) \rangle_H \]

Dot product between feature derivatives:

\[ \left\langle \frac{\partial}{\partial x_i} \varphi(X), \frac{\partial}{\partial x_j} \varphi(X') \right\rangle_H = \frac{\partial^2}{\partial x_i \partial x_{d+j}} k(X, X') \]

By representer theorem:

\[ f_n^* = \sum_{\ell=1}^n \sum_{j=1}^d \beta_{\ell j} \frac{\partial \varphi(X_\ell)}{\partial x_j} + \alpha \frac{1}{n} \sum_{\ell=1}^n \sum_{j=1}^d \left( \partial_j k(X_\ell, \cdot) \partial_j \log q_0(X_\ell) + \partial_j^2 k(X_\ell, \cdot) \right) \]

\[ \xi \]
A kernel solution

The RKHS solution

\[ f_n^*(x) = \alpha \hat{\xi}(x) + \sum_{\ell=1}^{n} \sum_{j=1}^{d} \beta_{\ell j} \frac{\partial k(x, X_\ell)}{\partial x_j} \]

Need to solve a linear system

\[
\begin{pmatrix} G_{XX} + n\lambda I \end{pmatrix} \beta_n^* = \frac{1}{\lambda} h_X
\]

\[
(h_X)_{(a-1)d+i}, := \langle \hat{\xi}, \partial_i k(x_a) \rangle
\]

Very costly in high dimensions!
The Nystrom approximation
Nystrom approach for efficient solution

- Find best estimator $f_{n,m}^*$ in $\mathcal{H}_Y := \text{span} \{ \partial_i k(y_a, \cdot) \}_{a \in [m], i \in [d]}$, where $y_a \in \{x_i\}_{i=1}^n$ chosen at random.

- Nystrom solution:

$$\beta_{n,m}^* = - \left( \frac{1}{n} B_{XY} B_{XY} + \lambda G_{YY} \right)^\dagger h_Y$$

Solve in time $O(n m^2 d^3)$, evaluate in time $O(md)$.

- Sill cubic in $d$, but similar results if we take a random dimension per datapoint.
Consistency: original solution

Define $C$ as the covariance between feature derivatives. Then from [Sriperumbudur et al. JMLR (2017)]

Rates of convergence: Suppose

- $f_0 \in \mathcal{R}(C^\beta)$ for some $\beta > 0$.
- $\lambda = n^{-\max\left\{\frac{1}{3}, \frac{1}{2(\beta+1)}\right\}}$ as $n \to \infty$.

Then

$$D_F(p_0, p_{f_n}) = O_{p_0} \left( n^{-\min\left\{\frac{2}{3}, \frac{\beta}{2(\beta+1)}\right\}} \right)$$
**Consistency: original solution**

Define $C$ as the covariance between feature derivatives. Then from [Sriperumbudur et al. JMLR (2017)]

- **Rates of convergence:** Suppose
  - $f_0 \in \mathcal{R}(C^\beta)$ for some $\beta > 0$.
  - $\lambda = n^{-\max\left\{\frac{1}{3}, \frac{1}{2(\beta+1)}\right\}}$ as $n \to \infty$.

  Then
  $$D_F(p_0, p_{f_n}) = O_{p_0} \left( n^{-\min\left\{\frac{2}{3}, \frac{\beta}{2(\beta+1)}\right\}} \right)$$

- **Convergence in other metrics:** KL, Hellinger, $L_r, 1 < r < \infty$. 

Consistency: Nystrom solution

Define $C$ as the covariance between feature derivatives.

- Suppose
  - $f_0 \in \mathcal{R}(C^\beta)$ for some $\beta > 0$.
  - Number of subsampled points $m = \Omega(n^\theta \log n)$ for $\theta = (\min(2\beta, 1) + 2)^{-1} \in \left[\frac{1}{3}, \frac{1}{2}\right]$
  - $\lambda = n^{-\max\left\{\frac{1}{3}, \frac{1}{2(\beta+1)}\right\}}$ as $n \to \infty$.

- Then
  $$D_F(p_0, p_{fn,m}) = O_{p_0} \left(n^{-\min\left\{\frac{2}{3}, \frac{\beta}{2(\beta+1)}\right\}}\right)$$
Define $C$ as the covariance between feature derivatives.

- Suppose
  - $f_0 \in \mathcal{R}(C^\beta)$ for some $\beta > 0$.
  - Number of subsampled points $m = \Omega(n^\theta \log n)$ for $\theta = (\min(2\beta, 1) + 2)^{-1} \in \left[\frac{1}{3}, \frac{1}{2}\right]$.
  - $\lambda = n^{-\max\left\{\frac{1}{3}, \frac{1}{2(\beta+1)}\right\}}$ as $n \to \infty$.

- Then
  $$D_F(p_0, p_{f_n,m}) = O_{p_0} \left(n^{-\min\left\{\frac{2}{3}, \frac{\beta}{2(\beta+1)}\right\}}\right)$$

- Convergence in other metrics: KL, Hellinger, $L_r$, $1 < r < \infty$. Same rate but saturates sooner.
  - Original (all samples) KL saturates at $O_{p_0} \left(n^{-\frac{1}{2}}\right)$
  - Nystrom saturates at $O_{p_0} \left(n^{-\frac{1}{3}}\right)$
A competing method: denoising autoencoder

Train a denoising autoencoder with Gaussian noise $\sigma$

Normalized reconstruction error estimates the score:

$$\frac{r_\sigma(x) - x}{\sigma} \rightarrow \nabla_x \log p_0(x)$$

- $r_\sigma(x)$ is reconstruction of noisy $x$ via encoder/decoder

Requirements for consistency: autoencoder has infinite capacity and is at global optimum

In practice: $\sigma$ is like a bandwidth, have to tune it
A competing method: denoising autoencoder

Train a denoising autoencoder with Gaussian noise $\sigma$

Normalized reconstruction error estimates the score:

$$\frac{r_\sigma(x) - x}{\sigma} \rightarrow \nabla_x \log p_0(x)$$

- $r_\sigma(x)$ is reconstruction of noisy $x$ via encoder/decoder

Requirements for consistency: autoencoder has infinite capacity and is at global optimum

In practice: $\sigma$ is like a bandwidth, have to tune it
Experimental results: ring

Sample:

Score:
Experimental results: comparison with autoencoder

Comparison with regularized auto-encoders [Alain and Bengio (JMLR, 2014)]

n=500 training points
Experimental results: grid of Gaussians

Sample:

Score:
Experimental results: comparison with autoencoder

- Comparison with regularized auto-encoders [Alain and Bengio (JMLR, 2014)]
- n=500 training points
The kernel conditional exponential family
The kernel conditional exponential family

- Can we take advantage of the graphical structure of \((X_1, ..., X_d)\)?
- Start from a general factorization of \(P\)

\[
P(X_1, ..., X_d) = \prod_i P(X_i \mid \underbrace{X_{\pi(i)}}_{\text{parents of } X_i})
\]

- Estimate each factor independently
Kernel conditional exponential family

General definition, kernel conditional exponential family

[Smola and Canu, 2006]

\[ p_f(y|x) = e^{\langle f, \psi(x,y) \rangle_H - A(f,x)} q_0(y) \]

\[ A(f, x) = \log \int q_0(y) e^{\langle f, \psi(x,y) \rangle_H} dy \]

(joint feature map \( \psi(x, y) \))
Kernel conditional exponential family

Our definition, kernel conditional exponential family:

\[ p_f(y|x) = e^{\langle f_x, \phi(y) \rangle_G - A(f,x)} q_0(y) \]

\[ A(f, x) = \log \int q_0(y) e^{\langle f_x, \phi(y) \rangle_G} \]

linear in the sufficient statistic \( \phi(y) \in G \).
Kernel conditional exponential family

Our definition, kernel conditional exponential family:

\[ p_f(y|x) = e^{\langle f_x, \phi(y) \rangle} q_0(y) - A(f, x) \]

\[ A(f, x) = \log \int q_0(y) e^{\langle f_x, \phi(y) \rangle} \]

linear in the sufficient statistic \( \phi(y) \in G \).

What would be the joint RKHS feature map \( \psi(x, y) \)?
Kernel conditional exponential family

What does the joint RKHS look like?  [Micchelli and Pontil, (2005)]

\[ \langle f_x, \phi(y) \rangle_{H} = \langle \Gamma_x f, \phi(y) \rangle_{G} = \langle f, \Gamma_x \phi(y) \rangle_{\mathcal{H}} \]

\[ \psi(x, y) \]
Kernel conditional exponential family

What does the joint RKHS look like? [Micchelli and Pontil, (2005)]

\[
\langle f_x, \phi(y) \rangle_g \\
= \langle \Gamma_x f, \phi(y) \rangle_g \\
= \langle f, \Gamma_x \phi(y) \rangle_{\mathcal{H}} \\
\psi(x,y)
\]

\[\Gamma_x : \mathcal{H} \to \mathcal{G} \text{ a linear operator evaluating } f \text{ at } x\]
Kernel conditional exponential family

What does the joint RKHS look like? [Micchelli and Pontil, (2005)]

\[
\langle f_x, \phi(y) \rangle_g \\
= \langle \Gamma^*_x f, \phi(y) \rangle_g \\
= \langle f, \Gamma_x \phi(y) \rangle_H \\
\underbrace{\psi(x,y)}_{\psi(x,y)}
\]

- \( \Gamma_x : G \rightarrow H \) is a linear operator.
- The feature map \( \psi(x, y) := \Gamma_x \phi(y) \)
Kernel conditional exponential family

What does the joint RKHS look like? [Micchelli and Pontil, (2005)]

\[
\langle f_x, \phi(y) \rangle_G = \langle \Gamma^*_x f, \phi(y) \rangle_G = \langle f, \Gamma_x \phi(y) \rangle_H \psi(x,y)
\]

- \( \Gamma_x : G \to \mathcal{H} \) is a linear operator.
- The feature map \( \psi(x, y) := \Gamma_x \phi(y) \)
- Simplest case:
  \( \Gamma_x = I_G k(x, \cdot) \) and \( \Gamma_x \phi(y) = \phi(y) k(x, \cdot) \)
What is our loss function?

The obvious approach: minimise

$$D_F [p_0(x)p_0(y|x)||p_0(x)p_f(y|x)]$$

Problem: the expression still contains $\int p_0(y|x)dy$. 
What is our loss function?

The obvious approach: minimise

\[ \mathcal{D}_F [p_0(x)p_0(y|x)||p_0(x)p_f(y|x)] \]

Problem: the expression still contains \( \int p_0(y|x)dy \).

Our loss function:

\[ \tilde{\mathcal{D}}_F(p_0, p_f) := \int \mathcal{D}_F(p_0(y|x)||p_f(y|x))\pi(x)\,dx \]

for some \( \pi(x) \) that includes the support of \( p(x) \).
Finite sample estimate of the conditional density

Use the simplest operator-valued RKHS $\Gamma_x = I_G k(x, \cdot)$.

$$
\Gamma_x : \mathcal{G} \rightarrow \mathcal{H}
$$

$$
\Gamma_x \phi(y) \mapsto \phi(y) k(x, \cdot)
$$
Finite sample estimate of the conditional density

Use the simplest operator-valued RKHS $\Gamma_x = I_G k(x, \cdot)$.

$$\Gamma_x : G \rightarrow \mathcal{H}$$
$$\Gamma_x \phi(y) \mapsto \phi(y) k(x, \cdot)$$

Solution:

$$f_n^*(y|x) = \sum_{b=1}^{n} \sum_{i=1}^{d} \beta_{(b,i)} k(x, X_b) \partial_i \kappa(y, Y_b) + \alpha \xi$$

where

$$\beta_n^* = -\frac{1}{\lambda} (G + n\lambda I)^{-1} h$$

$$(G)_{(a,i),(b,j)} = k(X_a, X_b) \partial_i \partial_j + d \kappa(Y_a, Y_b),$$

and $\langle \phi(y), \phi(y') \rangle_G = \kappa(y, y')$. 

Expected conditional score: a failure case

- $P(Y|X = 1)$
- $P(Y|X = -1)$
- $P(Y) = \frac{1}{2}(P(Y|X = 1) + P(Y|X = -1))$

$\tilde{D}_F(p(y|x), p(y)) = 0$
Expected conditional score: a failure case

- $P(Y|X = 1)$
- $P(Y|X = -1)$
- $P(Y) = \frac{1}{2}(P(Y|X = 1) + P(Y|X = -1))$

$\tilde{D}_F(p(y|x), p(y)) = 0$
Expected conditional score: a failure case

- \( P( Y|X = 1) \)
- \( P( Y|X = -1) \)
- \( P(Y) = \frac{1}{2}(P(Y|X = 1) + P(Y|X = -1)) \)

\[ \tilde{D}_F(p(y|x), \underbrace{p(y)}_{\text{model}}) = 0 \]
Expected conditional score: a failure case

- $P(Y|X = 1)$
- $P(Y|X = -1)$
- $P(Y) = \frac{1}{2}(P(Y|X = 1) + P(Y|X = -1))$

\[ \tilde{D}_F(p(y|x), p(y)) = 0 \]
Expected conditional score: a failure case

Why does it fail? Recall

\[ \tilde{D}_F(p_0(y|x), p_f(y|x)) := \int \pi(x) D_F(p_0(y|x), p_f(y|x)) \, dx \]

Note that

\[ D_F\left(\underbrace{p(y|x = 1)}_{\text{target}}, \underbrace{p(y)}_{\text{model}}\right) = \int \left| \nabla_y \log p(y|x = 1) - \nabla_y p(y) \right|^2 \, dy \]

Model \( p(y) \) puts mass where target conditional \( p(y|x = 1) \) has no support.

- Care needed when this failure mode approached!
Unconditional vs conditional model in practice

- **Red Wine**: Physiochemical measurements on wine samples.
- **Parkinsons**: Biomedical voice measurements from patients with early stage Parkinson’s disease.

<table>
<thead>
<tr>
<th></th>
<th>Parkinsons</th>
<th>Red Wine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension</td>
<td>15</td>
<td>11</td>
</tr>
<tr>
<td>Samples</td>
<td>5875</td>
<td>1599</td>
</tr>
</tbody>
</table>
Unconditional vs conditional model in practice

- **Red Wine**: Physiochemical measurements on wine samples.
- **Parkinsons**: Biomedical voice measurements from patients with early stage Parkinson’s disease.

Comparison with

- **LSCDE model**: with consistency guarantees [Sugiyama et al., (2010)]
- **RNADE model**: mixture models with deep features of parents, no guarantees [Uria et al. (2016)]
Unconditional vs conditional model in practice

- **Red Wine**: Physiochemical measurements on wine samples.
- **Parkinsons**: Biomedical voice measurements from patients with early stage Parkinson’s disease.

Comparison with

- LSCDE model: with consistency guarantees [Sugiyama et al., (2010)]
- RNADE model: mixture models with deep features of parents, no guarantees [Uria et al. (2016)]

**Negative log likelihoods** (smaller is better, average over 5 test/train splits)

<table>
<thead>
<tr>
<th></th>
<th>Parkinsons</th>
<th>Red wine</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCEF</td>
<td>2.86 ± 0.77</td>
<td>11.8 ± 0.93</td>
</tr>
<tr>
<td>LSCDE</td>
<td>15.89 ± 1.48</td>
<td>14.43 ± 1.5</td>
</tr>
<tr>
<td>NADE</td>
<td>3.63 ± 0.0</td>
<td>9.98 ± 0.0</td>
</tr>
</tbody>
</table>
Results: unconditional model

Red Wine Data

Parkinsons Data

\[
\begin{array}{c}
\text{Data} \\
\text{KEF}
\end{array}
\]

\[
\begin{array}{c}
\text{Data} \\
\text{KEF}
\end{array}
\]
Results: conditional model

Red Wine

Parkinsons

-6 −4 −2 0 2 4 x 6
−6
−4
−2
0
2
4
6

Red Wine Data
KCEF

Parkinsons Data
KCEF

39/66
Deep kernel infinite exponential models
"Combining a deep architecture with a kernel machine that takes the higher-level learned representation as input can be quite powerful."
"Combining a deep architecture with a kernel machine that takes the higher-level learned representation as input can be quite powerful."

Y. Bengio and Y. LeCun (2007)
The case for nonstationary (learned) kernels

Stationary kernels, nonstationary target:
The case for nonstationary (learned) kernels

Nonstationary kernels, nonstationary target:
The model class

Nonstationary kernels, nonstationary target:
Given a dataset $D := \{x_n\}_{n=1}^N$, empirical score matching loss is

$$\hat{J}(p_\theta, D) := \frac{1}{N} \sum_{n=1}^N \sum_{d=1}^D \left[ \partial_d^2 \log \tilde{p}_f(x_n) + \frac{1}{2} (\partial_d \log \tilde{p}_f(x_n))^2 \right]$$

The model has a natural parameter $f$ and sufficient statistic $k(x, \cdot)$:

$$\tilde{p}_f(x) = \exp(f(x)) q_0(x) = \exp(\langle f, k(x, \cdot) \rangle_\mathcal{H}) q_0(x).$$

Define a “lite” model of the form:

$$f_{\alpha, z}^k := \sum_{m=1}^M \alpha_m k_w(z_m, \cdot)$$

where $w$ are the kernel parameters (next slide).
Kernel design

Kernel of the form:

\[
k_w(x, y) = \sum_{r=1}^{R} \rho_r \exp \left( -\frac{1}{2\sigma^2_r} \| \phi_{wr}(x) - \phi_{wr}(y) \|^2 \right)
\]

\(\phi_{wr}\) are made up of \(L = 3\) fully connected layers.

- For \(L > 1\), skip connection directly to the top layer (\(L > 3\) hard to train due to second derivatives)
- Softplus nonlinearity, \(\log(1 + \exp(x))\): model is twice-differentiable, score well-defined.
- Same architecture and a linear kernel: performance was much worse.
The “lite” model

Regularised loss to fit model $\tilde{p}^k_{\alpha, z}$:

$$\hat{J}(f^k_{\alpha, z}, \lambda, \mathcal{D}) = \hat{J}(\tilde{p}^k_{\alpha, z}, \mathcal{D}) + \frac{\lambda_\alpha}{2} ||\alpha||^2 + \frac{\lambda_C}{2N} \sum_{n=1}^{N} \sum_{d=1}^{D} \left[ \partial^2_d \log \tilde{p}^k_{\alpha, z}(x_n) \right]^2$$

Comparison to earlier exponential family loss:

- The regulariser $\frac{\lambda_\alpha}{2} ||\alpha||^2$ is essential.
- Earlier work: primarily regularized with $\lambda_H ||f^k_{\alpha, z}||^2_{\mathcal{H}}$. As we change $k$, however, $||f||_{\mathcal{H}}$ changes meaning.
- The “curvature” term $\lambda_C \sum_{n=1}^{N} \sum_{d=1}^{D} \left[ \partial^2_d \log \tilde{p}^k_{\alpha, z}(x_n) \right]^2$ is from Kingma and LeCun (2010), but it rarely makes a difference (small improvement on one dataset).
The “lite” model

Regularised loss to fit model $\tilde{p}_\alpha^k, z$:

$$
\hat{J}(f_\alpha^k, \lambda, D) = \underbrace{\hat{J}(\tilde{p}_\alpha^k, z, D)}_{\text{unreg. loss}} + \underbrace{\frac{\lambda_\alpha}{2} \|\alpha\|^2}_{\ell_2 \text{ reg.}} + \underbrace{\frac{\lambda_c}{2N} \sum_{n=1}^N \sum_{d=1}^D \left[ \partial_d^2 \log \tilde{p}_\alpha^k, z(x_n) \right]^2}_{\text{curvature reg.}}
$$

Comparison to earlier exponential family loss:

- The regulariser $\frac{\lambda_\alpha}{2} \|\alpha\|^2$ is essential.
- Earlier work: primarily regularized with $\lambda_H \|f_\alpha^k, z\|^2_{\mathcal{H}}$. As we change $k$, however, $\|f\|_{\mathcal{H}}$ changes meaning.
- The “curvature” term $\lambda_c \sum_{n=1}^N \sum_{d=1}^D \left[ \partial_d^2 \log \tilde{p}_\alpha^k, z(x_n) \right]^2$ is from Kingma and LeCun (2010), but it rarely makes a difference (small improvement on one dataset).
The weights $\alpha$ are solutions to a linear system

Nonstationary kernels, nonstationary target:

Minimiser of $\tilde{J}(f_{\alpha,z}^k, \lambda, \mathcal{D})$ obtained in $O(M^2 ND + M^3)$ time,

$$\alpha(\lambda, k, z, \mathcal{D}) = - (G + \lambda_{\alpha} I + \lambda_C U)^{-1} b$$

$$G_{m,m'} = \frac{1}{N} \sum_{n=1}^{N} \sum_{d=1}^{D} \partial_d k(x_n, z_m) \partial_d k(x_n, z_{m'})$$

$$U_{m,m'} = \frac{1}{N} \sum_{n=1}^{N} \sum_{d=1}^{D} \partial_d^2 k(x_n, z_m) \partial_d^2 k(x_n, z_{m'})$$

$$b_m = \frac{1}{N} \sum_{n=1}^{N} \sum_{d=1}^{D} \partial_d^2 k(x_n, z_m) + \partial_d \log q_0(x_n) \partial_d k(x_n, z_m) + \lambda_C \partial_d^2 \log q_0(x_n) \partial_d^2 k(x_n, z_m).$$
The algorithm

The challenge: we are optimising over many things:

- the locations of the inducing points, $z$
- The parameters $w$ of the convolutional features $\phi$, including kernel weights $\rho_r$.
- The regularisation coefficients $\lambda_C$ and $\lambda_\alpha$
- The coefficients $\alpha$ themselves.

What doesn’t work: joint optimisation over $w, \alpha, \lambda$. Kernels collapse to delta functions.
The algorithm

The challenge: we are optimising over many things:

- the locations of the inducing points, \( z \)
- The parameters \( w \) of the convolutional features \( \phi \), including kernel weights \( \rho_r \).
- The regularisation coefficients \( \lambda_C \) and \( \lambda_\alpha \)
- The coefficients \( \alpha \) themselves.

What doesn’t work: joint optimisation over \( w, \alpha, \lambda \). Kernels collapse to delta functions.

We split the data: \( \mathcal{D} = \{ \mathcal{D}_1, \mathcal{D}_2 \} \).

- **Stage 1:** \( \mathcal{D}_2 \) is used to monitor convergence while optimising \( w \) and \( z \).
- **Stage 2:** \( \mathcal{D}_2 \) is used to define a validation loss on which to optimise \( \alpha \) and \( \lambda \).
Stage 1: learning $w$ and $z$

Fitting regulariser, inducing point:
While $\hat{J}(\tilde{p}_{k_w}^\alpha(\lambda, k_w, z, D_1), z, D_2)$ still improving do

- Sample disjoint data subsets $D_t, D_v \subset D_1$
- Express natural parameter using inducing points,
  $f(\cdot) = \sum_{m=1}^{M} \alpha_m(\lambda, k_w, z, D_t) k_w(z_m, \cdot)$
  - $\alpha_m$ solved on training data $D_t$.
- Define unregularised validation loss on $D_v$:
  $\hat{J} = \frac{1}{|D_v|} \sum_{n=1}^{D_v} \sum_{d=1}^{D} \left[ \partial_d^2 f(x_n) + \frac{1}{2}(\partial_d f(x_n))^2 \right]$

- Take SGD steps in $\hat{J}$ for $w$, $\lambda$, and optionally $z$. 
Stage 2: refinement of $\lambda$

Once kernel parameters $w$ and inducing points learned, refine solution on $\alpha$ and $\lambda$:

While $\hat{J}(\tilde{p}\alpha^{k_w}_{\lambda,k_w,z,D_1},z,D_2)$ still improving do

- Express natural parameter using inducing points, this time solving on all $D_1$, $f(\cdot) = \sum_{m=1}^{M} \alpha_m(\lambda, k_w, z, D_1) k_w(z_m, \cdot)$
- Define unregularised validation loss on $D_2$,

$$
\hat{J} = \frac{1}{|D_2|} \sum_{n=1}^{|D_2|} \sum_{d=1}^{D} \left[ \partial_d^2 f(x_n) + \frac{1}{2}(\partial_d f(x_n))^2 \right]
$$

- Take SGD steps in $\hat{J}$ for $\lambda$ only.
What works, what doesn’t work, and why

“The usual suspects”:

Funnel -3.44  Banana -3.49  Ring -3.25  Square -3.58  Cosine -3.49
What works, what doesn’t work, and why

Learned kernels vs fixed kernels:

KEF-G

-3.48, 0.21
-3.53, 0.05
-3.27, 0.16
-3.64, 4.58
-3.63, 1.27

DKEF-G-15
What works, what doesn’t work, and why

MADE with mixture of Gaussians:

Definition of MADE (Masked Autoencoder for Distribution Estimation):

\[ p(x) := \prod_{d=1}^{D} p(x_d|x_{<d}), \]

each probability a mixture of Gaussians with parameters deep features of \( x_{<d} \) (this variant called MADE-MOG).
What works, what doesn’t work, and why

**MAF** (masked autoregressive flow)

![MAF images]

Definition of masked autoregressive flow:

\[
p(x_i|x_{1:i-1}) = \mathcal{N}(x_i|\mu_i, (\exp \alpha_i)^2)
\]

\[
\mu_i = f_{\mu_i}(x_{1:i-1})
\]

\[
\alpha_i = f_{\alpha_i}(x_{1:i-1})
\]

\[
x_i = u_i \exp(\alpha_i) + \mu_i
\]

Depth: output of model is used as noise input \( u_i \) for the next layer.
What works, what doesn’t work, and why

MAF (masked autoregressive flow) with mixture of Gaussians

**Definition** of masked autoregressive flow:

\[
p(x_i|x_{1:i-1}) = \mathcal{N}(x_i|\mu_i, (\exp \alpha_i)^2)\]

\[
\mu_i = f_{\mu_i}(x_{1:i-1})
\]

\[
\alpha_i = f_{\alpha_i}(x_{1:i-1})
\]

\[
x_i = u_i \exp(\alpha_i) + \mu_i
\]

Depth: output of model is used as noise input \( u_i \) for the next layer.

**MAF-MOG**: stacked five-deep, using MADE-MOG with \( C = 10 \) Gaussian components as base density \( u_i \).
Two simple datasets

Disconnected mixture of two Gaussians, and bullseye:
How does MAF do?

Disconnected mixture of two Gaussians, and bullseye:
How does kernel exponential family do?

Disconnected mixture of two Gaussians, and bullseye:

KEF-G

-2.20, 0.035

-4.10, 0.94

-3.85, 0.8

DKEF-G-15

-2.37, 0.018

-10.67, 0.45

-3.86, 0.88
Solutions, kernel Stein discrepancy and log likelihood

Once kernel parameters $w$ and inducing points learned, refine solution on $\alpha$ and $\lambda$:
Application: adaptive Hamiltonian Monte Carlo
Bayesian Gaussian process classification

Our case: target $\pi(\cdot)$ and log gradient not computable - Pseudo-Marginal MCMC

When is likelihood not computable?

- GPC model: latent process $f$, labels $y$, (with covariate matrix $X$), and hyperparameters $\theta$:
  \[
p(f, y, \theta) = p(\theta)p(f|\theta)p(y|f)
  \]
  $f|\theta \sim \mathcal{N}(0, K_{\theta})$ GP with covariance $K_{\theta}$

- Automatic Relevance Determination (ARD) covariance:
  \[
  (K_{\theta})_{ij} = \kappa(x_i, x_j'|\theta) = \exp \left( -\frac{1}{2} \sum_{s=1}^{d} \frac{(x_{i,s} - x_{j,s}')^2}{\exp(\theta_s)} \right)
  \]

- $p(y|f) = \prod_{i=1}^{n} p(y_i|f(x_i))$ where
  \[
p(y_i|f(x_i)) = (1 - \exp(-y_if(x_i)))^{-1}, \quad y_i \in \{-1, 1\}.
  \]
Bayesian Gaussian process classification

Our case: target \( \pi(\cdot) \) and log gradient not computable -
Pseudo-Marginal MCMC

When is likelihood not computable?

- GPC model: latent process \( f \), labels \( y \), (with covariate matrix \( X \)), and hyperparameters \( \theta \):
  \[
p(f, y, \theta) = p(\theta)p(f|\theta)p(y|f)
\]
  \( f|\theta \sim \mathcal{N}(0, \mathcal{K}_\theta) \) GP with covariance \( \mathcal{K}_\theta \)

- Automatic Relevance Determination (ARD) covariance:
  \[
  (\mathcal{K}_\theta)_{ij} = \kappa(x_i, x_j'|\theta) = \exp \left( -\frac{1}{2} \sum_{s=1}^{d} \frac{(x_{i,s} - x'_{j,s})^2}{\exp(\theta_s)} \right)
  \]

- \( p(y|f) = \prod_{i=1}^{n} p(y_i|f(x_i)) \) where
  \[
p(y_i|f(x_i)) = (1 - \exp(-y_i f(x_i)))^{-1}, \quad y_i \in \{-1, 1\}.
  \]
Bayesian Gaussian process classification

Our case: target \( \pi(\cdot) \) and log gradient not computable - Pseudo-Marginal MCMC

When is likelihood not computable?

- GPC model: latent process \( f \), labels \( y \), (with covariate matrix \( X \)), and hyperparameters \( \theta \):

\[
p(f, y, \theta) = p(\theta)p(f|\theta)p(y|f)
\]

\( f|\theta \sim \mathcal{N}(0, K_\theta) \) GP with covariance \( K_\theta \)

- Automatic Relevance Determination (ARD) covariance:

\[
(K_\theta)_{ij} = \kappa(x_i, x_j'|\theta) = \exp \left( -\frac{1}{2} \sum_{s=1}^{d} \frac{(x_{i,s} - x'_{j,s})^2}{\exp(\theta_s)} \right)
\]

- \( p(y|f) = \prod_{i=1}^{n} p(y_i|f(x_i)) \) where

\[
p(y_i|f(x_i)) = (1 - \exp(-y_i f(x_i)))^{-1}, \quad y_i \in \{-1, 1\}.
\]
Bayesian Gaussian process classification

Our case: target $\pi(\cdot)$ and log gradient not computable - Pseudo-Marginal MCMC

When is likelihood not computable?

- GPC model: latent process $f$, labels $y$, (with covariate matrix $X$), and hyperparameters $\theta$:

$$p(f, y, \theta) = p(\theta)p(f|\theta)p(y|f)$$

$f|\theta \sim \mathcal{N}(0, \mathcal{K}_\theta)$ GP with covariance $\mathcal{K}_\theta$

- Automatic Relevance Determination (ARD) covariance:

$$(\mathcal{K}_\theta)_{ij} = \kappa(x_i, x'_j|\theta) = \exp \left( -\frac{1}{2} \sum_{s=1}^{d} \frac{(x_{i,s} - x'_{j,s})^2}{\exp(\theta_s)} \right)$$

- $p(y|f) = \prod_{i=1}^{n} p(y_i|f(x_i))$ where

$$p(y_i|f(x_i)) = \frac{1}{1 + \exp(-y_if(x_i))}, \quad y_i \in \{-1, 1\}.$$
Bayesian Gaussian process classification

Example: when is target not computable?

- Gaussian process classification, latent process \( f \)

\[
p(\theta|y) \propto p(\theta)p(y|\theta) = p(\theta) \int p(f|\theta)p(y|f, \theta) df =: \pi(\theta)
\]

... but cannot integrate out \( f \)

- Metropolis Hastings ratio:

\[
\alpha(\theta, \theta') = \min \left\{ 1, \frac{p(\theta')p(y|\theta') q(\theta'|\theta)}{p(\theta)p(y|\theta) q(\theta|\theta')} \right\}
\]

- Pseudo-Marginal MCMC: unbiased estimate of \( p(y|\theta) \) via importance sampling: [Filippone & Girolami, (2013)]

\[
\hat{p}(\theta|y) \propto p(\theta)\hat{p}(y|\theta) \approx p(\theta) \frac{1}{n_{\text{imp}}} \sum_{i=1}^{n_{\text{imp}}} p(y|f^{(i)}) \frac{p(f^{(i)}|\theta)}{Q(f^{(i)})}
\]
Bayesian Gaussian process classification

Example: when is target not computable?

- Gaussian process classification, latent process $f$

$$p(\theta|y) \propto p(\theta)p(y|\theta) = p(\theta)\int p(f|\theta)p(y|f, \theta) df =: \pi(\theta)$$

... but cannot integrate out $f$

- Estimated MH ratio:

$$\alpha(\theta, \theta') = \min \left\{ 1, \frac{p(\theta')\hat{p}(y|\theta')q(\theta|\theta')}{p(\theta)\hat{p}(y|\theta)q(\theta'|\theta)} \right\}$$

- Replacing marginal likelihood $p(y|\theta)$ with unbiased estimate $\hat{p}(y|\theta)$ still results in correct invariant distribution [Beaumont (2003); Andrieu & Roberts (2009)]
Adaptive HMC

Sliced posterior over hyperparameters of a Gaussian Process classifier on UCI Glass dataset obtained using Pseudo-Marginal MCMC.

Can you learn an HMC sampler?
Basic adaptive Metropolis-Hastings

Sliced posterior over hyperparameters of a Gaussian Process classifier on UCI Glass dataset obtained using Pseudo-Marginal MCMC.

Significant improvements over random walk
Efficiency gains from approximate solution

HMC and acceptance rates for 90% quantiles

Acceptance rate

\( \begin{align*}
\text{full} & : 0.0, 0.2, 0.4, 0.6 \\
\text{nyst.} & : 0.0, 0.2, 0.4, 0.6 \\
\text{lite} & : 0.0, 0.2, 0.4, 0.6 \\
\text{dae} & : 0.0, 0.2, 0.4, 0.6
\end{align*} \)

\( m = 10 \), \( m = 100 \), \( m = 500 \)

Acceptance rate vs. \( m \) for different methods.
Co-authors

From Gatsby:
- Michael Arbel
- Kevin Li
- Heiko Strathmann
- Dougal Sutherland

External collaborators:
- Kenji Fukumizu
- Bharath Sriperumbudur

Questions?