Advances in kernel exponential families

Arthur Gretton

Gatsby Computational Neuroscience Unit,
University College London

Tenerife, 2018

Outline

Motivating application:

- Fast estimation of complex multivariate densities

The infinite exponential family:

- Multivariate Gaussian $\rightarrow$ Gaussian process
- Finite mixture model $\rightarrow$ Dirichlet process mixture model
- Finite exponential family $\rightarrow$???

Application:

- Adaptive HMC for Pseudo-Margial 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 $\to$ Gaussian process
- Finite mixture model $\to$ Dirichlet process mixture model
- Finite exponential family $\to$???

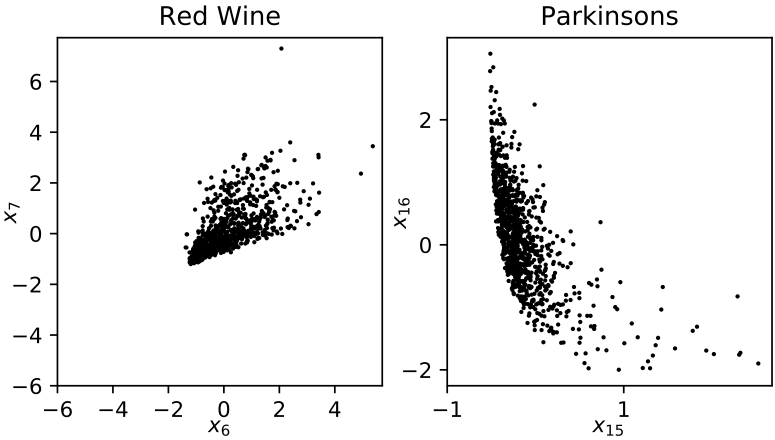
Application:

- Adaptive HMC for Pseudo-Margial 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:

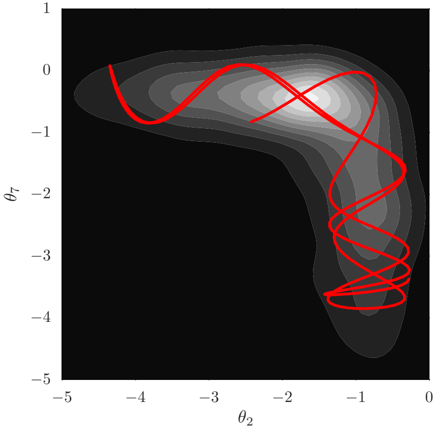
- Efficient computation and representation
- Statistical guarantees

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

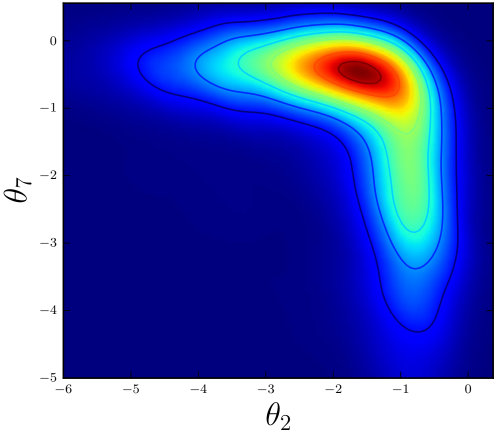
$$\frac{\partial K}{\partial p}\frac{\partial}{\partial x} - \frac{\partial U}{\partial x}\frac{\partial}{\partial p}$$

- 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 in $\mathbb{R}^d$

$$p(x) = \exp\left(\left\langle \underbrace{\eta}_{\substack{\text{natural}\\\text{parameter}}} , \underbrace{T(x)}_{\substack{\text{sufficient}\\\text{startistic}}} \right\rangle - \underbrace{A(\eta)}_{\substack{\text{log}\\\text{normaliser}}} \right) \underbrace{q_0(x)}_{\substack{\text{base}\\\text{measure}}}$$

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

Kernel exponential families [Canu and Smola (2006), Fukumizu (2009)] and their GP counterparts [Adams, Murray, MacKay (2009), Rasmussen(2003)]

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

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

Kernel exponential families [Canu and Smola (2006), Fukumizu (2009)] and their GP counterparts [Adams, Murray, MacKay (2009), Rasmussen(2003)]

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

where

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

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

- Example: Gaussian kernel, $T(x) = \left[\begin{array}{cc} x & x^2 \end{array} \right] = \varphi(x)$ and $k(x, y) = xy + x^2 y^2$

Fitting an infinite dimensional exponential family

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:

$$
\begin{aligned}
f_{ML} &= \arg\max_{f \in \mathcal{F}} \sum_{i=1}^{n} \log p_f(X_i) \\
&= \arg\max_{f \in \mathcal{F}} \sum_{i=1}^{n} f(X_i) - n \log \int e^{f(x)} q_0(x)\, dx.
\end{aligned}
$$

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

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

where $p_{f_{ML}} = \frac{d\mathbb{P}_{\mathrm{ML}}}{dx}$.

Ill posed for infinite dimensional $\varphi(x)$!

Score matching

Journal of Machine Learning Research 6 (2005) 695–708 Submitted 11/04; Revised 3/05; Published 4/05

Estimation of Non-Normalized Statistical Models by Score Matching

Aapo Hyvärinen AAPO.HYVARINEN@HELSINKI.FI
Helsinki Institute for Information Technology (BRU)
Department of Computer Science
FIN-00014 University of Helsinki, Finland

Editor: Peter Dayan

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 Ω: open subset of $\mathbb{R}^d$ with piecewise smooth boundary $\partial\Omega := \overline{\Omega} \backslash \Omega$,

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 $\partial^{\alpha,\alpha} k$ 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 $x \in \Omega$, $\|x\|_2 \to \infty$, $\forall i \in [d]$.

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

$$
\begin{aligned}
& D_F(p_0, p_f) \\
&= \frac{1}{2}\int_a^b p_0(x)\left(\frac{d\log p_0(x)}{dx} - \frac{d\log p_f(x)}{dx}\right)^2 dx \\
&= \frac{1}{2}\int_a^b p_0(x)\left(\frac{d\log p_0(x)}{dx}\right)^2 dx + \frac{1}{2}\int_a^b p_0(x)\left(\frac{d\log p_f(x)}{dx}\right)^2 dx \\
&\quad - \int_a^b p_0(x)\left(\frac{d\log p_f(x)}{dx}\right)\left(\frac{d\log p_0(x)}{dx}\right) dx
\end{aligned}
$$

Score matching: 1-D proof

$$
\begin{aligned}
& D_F(p_0, p_f) \\
&= \frac{1}{2}\int_a^b p_0(x)\left(\frac{d\log p_0(x)}{dx} - \frac{d\log p_f(x)}{dx}\right)^2 dx \\
&= \frac{1}{2}\int_a^b p_0(x)\left(\frac{d\log p_0(x)}{dx}\right)^2 dx + \frac{1}{2}\int_a^b p_0(x)\left(\frac{d\log p_f(x)}{dx}\right)^2 dx \\
&\quad - \int_a^b p_0(x)\left(\frac{d\log p_f(x)}{dx}\right)\left(\frac{d\log p_0(x)}{dx}\right) dx
\end{aligned}
$$

Final term:

$$
\begin{aligned}
& \int_a^b p_0(x)\left(\frac{d\log p_f(x)}{dx}\right)\left(\frac{d\log p_0(x)}{dx}\right) dx \\
&= \int_a^b p_0(x)\left(\frac{d\log p_f(x)}{dx}\right)\left(\frac{1}{p_0(x)}\frac{dp_0(x)}{dx}\right) dx \\
&= \left[\left(\frac{d\log p_f(x)}{dx}\right) p_0(x)\right]_a^b - \int_a^b p_0(x)\frac{d^2\log p_f(x)}{dx^2}.
\end{aligned}
$$

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:

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

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

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

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:

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

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

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

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 $p_t := p * N(0, tI_d)$ and $q_t := q * N(0, tI_d)$ 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*

$$\partial_i p_t(x) \log p_t(x) = o\left(\|x\|_2^\alpha\right), \ \partial_i p_t(x) \log q_t(x) = o\left(\|x\|_2^\alpha\right) \ \textit{and} \ \partial_i \log q_t(x) p_t(x) = o\left(\|x\|_2^\alpha\right)$$

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

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

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 $\lambda \|f\|_{\mathcal{H}}^2$ 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:

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

Thus

$$\frac{\partial}{\partial x} \log p_f(x) = \frac{\partial}{\partial x} \langle f, \varphi(x) \rangle_{\mathcal{H}} + \frac{\partial}{\partial x} \log q_0(x).$$

Kernel trick for derivatives: $\frac{\partial}{\partial x_i} f(X) = \left\langle f, \frac{\partial}{\partial x_i} \varphi(X) \right\rangle_{\mathcal{H}}$ Dot product between feature derivatives:

$$\left\langle \frac{\partial}{\partial x_i} \varphi(X), \frac{\partial}{\partial x_j} \varphi(X') \right\rangle_{\mathcal{H}} = \frac{\partial^2}{\partial x_i \partial x_{d+j}} k(X, X')$$

A kernel solution

Infinite exponential family:

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

Thus

$$\frac{\partial}{\partial x} \log p_f(x) = \frac{\partial}{\partial x} \langle f, \varphi(x) \rangle_{\mathcal{H}} + \frac{\partial}{\partial x} \log q_0(x).$$

Kernel trick for derivatives: $\frac{\partial}{\partial x_i} f(X) = \left\langle f, \frac{\partial}{\partial x_i} \varphi(X) \right\rangle_{\mathcal{H}}$ Dot product between feature derivatives:

$$\left\langle \frac{\partial}{\partial x_i} \varphi(X), \frac{\partial}{\partial x_j} \varphi(X') \right\rangle_{\mathcal{H}} = \frac{\partial^2}{\partial x_i \partial x_{d+j}} k(X, X')$$

By representer theorem:

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

A kernel solution

The RKHS solution

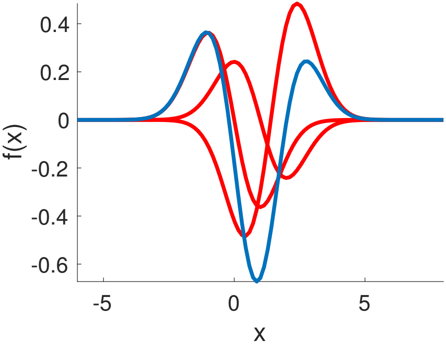
$$f_n^*(x) = \alpha\hat{\xi}(x) + \sum_{\ell=1}^{n}\sum_{j=1}^{d}\beta_{\ell j}\frac{\partial k(x, X_\ell)}{\partial x_j}$$

Need to solve a linear system

$$\Big(\underbrace{G_{XX}}_{nd\times nd} + n\lambda I\Big)\beta_n^* = \frac{1}{\lambda}h_X$$

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

Very costly in high dimensions!



The Nystrom approximation

Nystrom approach for efficient solution

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

$$\beta^*_{n,m} = -\left(\frac{1}{n} B_{XY}^\top \underbrace{B_{XY}}_{nd\times md} + \lambda \underbrace{G_{YY}}_{md\times md}\right)^\dagger h_Y$$

Solve in time $\mathcal{O}(nm^2d^3)$, evaluate in time $\mathcal{O}(md)$.

- 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

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

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

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

- 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

What Regularized Auto-Encoders Learn from the Data-Generating Distribution

Guillaume Alain GUILLAUME.ALAIN@UMONTREAL.CA
Yoshua Bengio YOSHUA.BENGIO@UMONTREAL.CA

- Train a denoising autoencoder with Gaussian noise σ
- Normalized reconstruction error estimates the score:

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

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

- Requirements for consistency: autoencoder has infinite capacity and is at global optimum
- In practice: σ is like a bandwidth, have to tune it

A competing method: denoising autoencoder

What Regularized Auto-Encoders Learn from the Data-Generating Distribution

Guillaume Alain GUILLAUME.ALAIN@UMONTREAL.CA
Yoshua Bengio YOSHUA.BENGIO@UMONTREAL.CA

- Train a denoising autoencoder with Gaussian noise σ
- Normalized reconstruction error estimates the score:

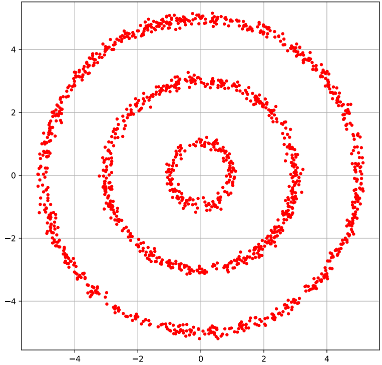
$$\frac{r_\sigma(x) - x}{\sigma} \to \nabla_x \log p_0(x)$$

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

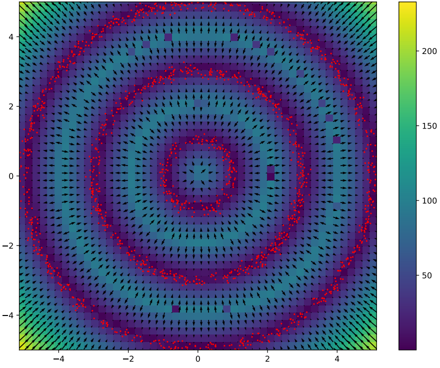
- Requirements for consistency: autoencoder has infinite capacity and is at global optimum
- In practice: σ is like a bandwidth, have to tune it

Experimental results: ring

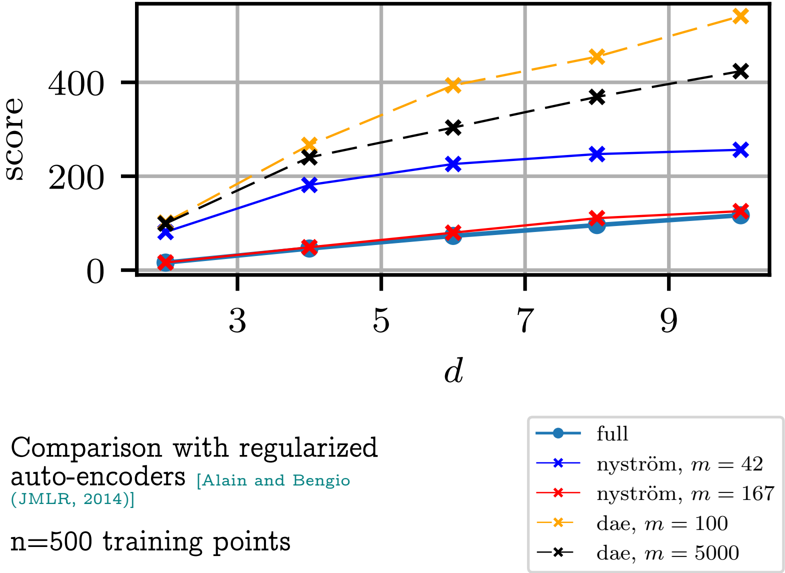
Sample:



Score:



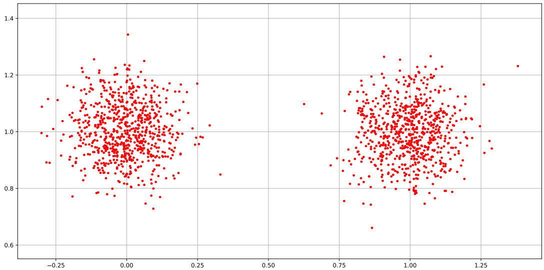
Experimental results: comparison with autoencoder



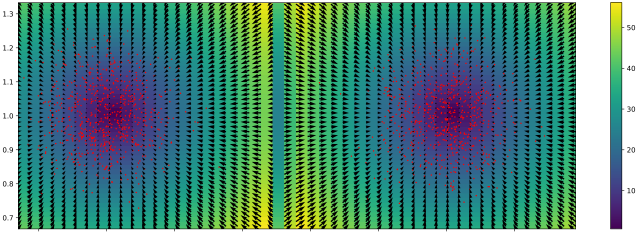
- Comparison with regularized auto-encoders [Alain and Bengio (JMLR, 2014)]
- n=500 training points

Experimental results: grid of Gaussians

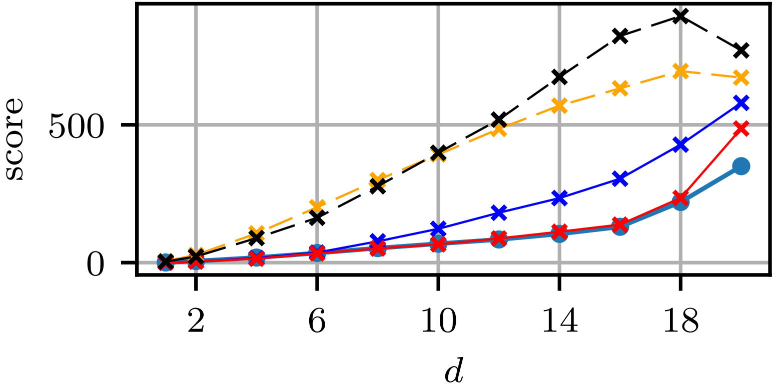
Sample:

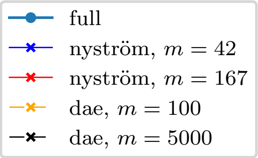


Score:



Experimental results: comparison with autoencoder





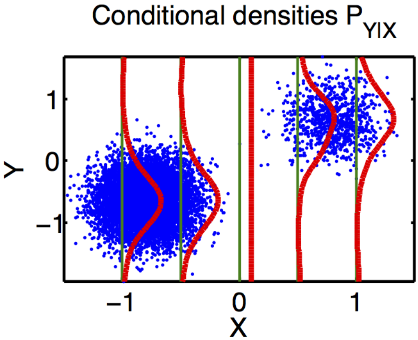
- Comparison with regularized auto-encoders [Alain and Bengio (JMLR, 2014)]
- n=500 training points

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

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



- Estimate each factor independently

Kernel conditional exponential family

General definition, kernel conditional exponential family

[Smola and Canu, 2006]

$$p_f(y|x) = e^{\langle f, \psi(x,y) \rangle_{\mathcal{H}} - A(f,x)} q_0(y) \qquad A(f,x) = \log \int q_o(y) e^{\langle f, \psi(x,y) \rangle_{\mathcal{H}}} dy$$

(joint feature map $\psi(x, y)$)

Kernel conditional exponential family

Our definition, kernel conditional exponential family:

$$p_f(y|x) = e^{\langle f_x, \phi(y) \rangle_{\mathcal{G}} - A(f,x)} q_0(y) \qquad A(f,x) = \log \int q_o(y) e^{\langle f_x, \phi(y) \rangle_{\mathcal{G}}}$$

linear in the sufficient statistic $\phi(y) \in \mathcal{G}$.

Kernel conditional exponential family

Our definition, kernel conditional exponential family:

$$p_f(y|x) = e^{\langle f_x, \phi(y) \rangle_{\mathcal{G}} - A(f,x)} q_0(y) \qquad A(f,x) = \log \int q_o(y) e^{\langle f_x, \phi(y) \rangle_{\mathcal{G}}}$$

linear in the sufficient statistic $\phi(y) \in \mathcal{G}$.

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)]

$$\langle f_x, \phi(y) \rangle_{\mathcal{G}}$$

$$= \langle \Gamma_x^* f, \phi(y) \rangle_{\mathcal{G}}$$

$$= \langle f, \underbrace{\Gamma_x \phi(y)}_{\psi(x,y)} \rangle_{\mathcal{H}}$$

Kernel conditional exponential family

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

$$
\begin{aligned}
&\langle f_x, \phi(y)\rangle_{\mathcal{G}} \\
&= \langle \Gamma_x^* f, \phi(y)\rangle_{\mathcal{G}} \\
&= \langle f, \underbrace{\Gamma_x \phi(y)}_{\psi(x,y)}\rangle_{\mathcal{H}}
\end{aligned}
$$

- $\Gamma_x^* : \mathcal{H} \to \mathcal{G}$ a linear operator evaluating f at x

Kernel conditional exponential family

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

$$
\begin{aligned}
&\langle f_x, \phi(y) \rangle_{\mathcal{G}} \\
&= \langle \Gamma_x^* f, \phi(y) \rangle_{\mathcal{G}} \\
&= \langle f, \underbrace{\Gamma_x \phi(y)}_{\psi(x,y)} \rangle_{\mathcal{H}}
\end{aligned}
$$

- $\Gamma_x : \mathcal{G} \to \mathcal{H}$ 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)]

$$\langle f_x, \phi(y)\rangle_{\mathcal{G}} = \langle \Gamma_x^* f, \phi(y)\rangle_{\mathcal{G}} = \langle f, \underbrace{\Gamma_x \phi(y)}_{\psi(x,y)}\rangle_{\mathcal{H}}$$

- $\Gamma_x : \mathcal{G} \to \mathcal{H}$ is a linear operator.
- The feature map $\psi(x, y) := \Gamma_x \phi(y)$
- Simplest case: $\Gamma_x = I_{\mathcal{G}} k(x, \cdot)$ and $\Gamma_x \phi(y) = \phi(y) k(x, \cdot)$

What is our loss function?

The obvious approach: minimise

$$D_F \left[p_0(x) p_0(y|x) \| p_0(x) p_f(y|x) \right]$$

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

What is our loss function?

The obvious approach: minimise

$$D_F\left[p_0(x)p_0(y|x)||p_0(x)p_f(y|x)\right]$$

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

Our loss function:

$$\widetilde{D}_F(p_0, p_f) := \int D_F(p_0(y|x)||p_f(y|x))\pi(x)\,dx$$

for some $\pi(x)$ that includes the support of $p(x)$.

Finite sample estimate of the conditional density

Use the simplest operator-valued RKHS $\Gamma_x = I_{\mathcal{G}} k(x, \cdot)$.

$$\begin{aligned} \Gamma_x &: \mathcal{G} \to \mathcal{H} \\ \Gamma_x \phi(y) &\mapsto \phi(y) k(x, \cdot) \end{aligned}$$

Finite sample estimate of the conditional density

Use the simplest operator-valued RKHS $\Gamma_x = I_{\mathcal{G}} k(x, \cdot)$.

$$\begin{aligned} \Gamma_x &: \mathcal{G} \to \mathcal{H} \\ \Gamma_x \phi(y) &\mapsto \phi(y) k(x, \cdot) \end{aligned}$$

Solution:

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

where

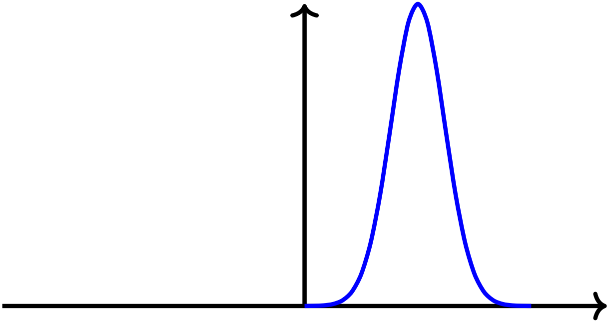
$$\beta_n^* = -\frac{1}{\lambda} (G + n\lambda I)^{-1} h$$

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

and $\langle \phi(y), \phi(y') \rangle_{\mathcal{G}} = \mathfrak{K}(y, y')$.

Expected conditional score: a failure case

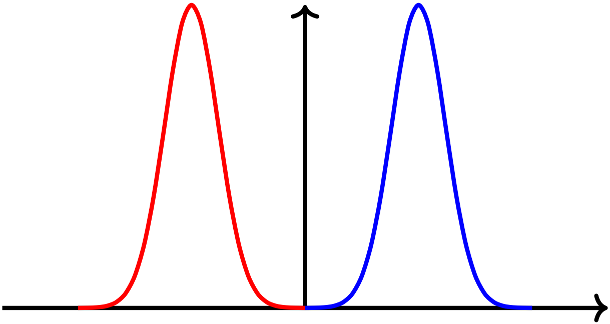
- $P(Y|X=1)$
- $P(Y|X=-1)$
- $P(Y) = \frac{1}{2}(P(Y|X=1) + P(Y|X=-1))$



$$\widetilde{D}_F(\underbrace{p(y|x)}_{\text{target}}, \underbrace{p(y)}_{\text{model}}) = 0$$

Expected conditional score: a failure case

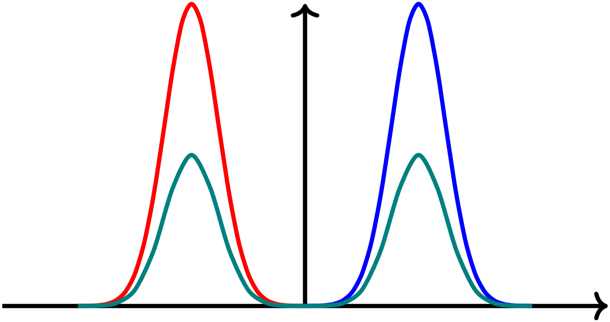
- $P(Y|X=1)$
- $P(Y|X=-1)$
- $P(Y) = \frac{1}{2}(P(Y|X=1) + P(Y|X=-1))$



$$\widetilde{D}_F(\underbrace{p(y|x)}_{\text{target}}, \underbrace{p(y)}_{\text{model}}) = 0$$

Expected conditional score: a failure case

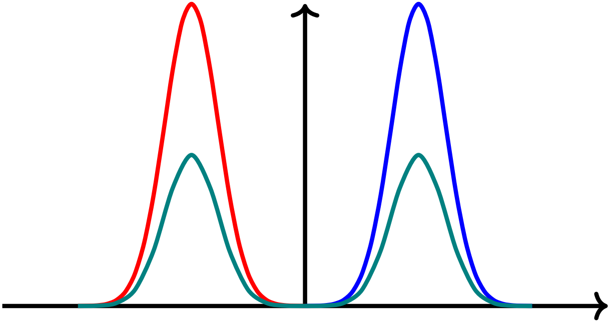
- $P(Y|X=1)$
- $P(Y|X=-1)$
- $P(Y) = \frac{1}{2}(P(Y|X=1) + P(Y|X=-1))$



$$\widetilde{D}_F(\underbrace{p(y|x)}_{\text{target}}, \underbrace{p(y)}_{\text{model}}) = 0$$

Expected conditional score: a failure case

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



$$\widetilde{D}_F(\underbrace{p(y|x)}_{\text{target}}, \underbrace{p(y)}_{\text{model}}) = 0$$

Expected conditional score: a failure case

Why does it fail? Recall

$$\widetilde{D}_F(p_0(y|x), p_f(y|x)) := \int \pi(x) D_F(p_0(y|x), p_f(y|x)) dx$$

Note that

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

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.

	Parkinsons	Red Wine
Dimension	15	11
Samples	5875	1599

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.

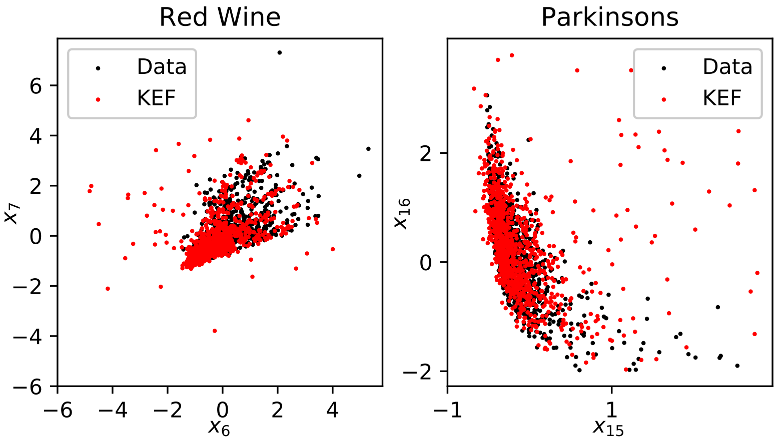
Comparison with

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

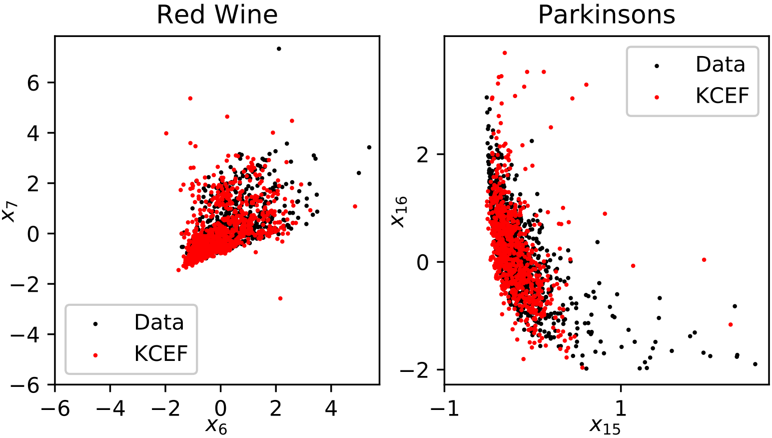
Negative log likelihoods (smaller is better, average over 5 test/train splits)

	Parkinsons	Red wine
KCEF	$\mathbf{2.86 \pm 0.77}$	11.8 ± 0.93
LSCDE	15.89 ± 1.48	14.43 ± 1.5
NADE	3.63 ± 0.0	$\mathbf{9.98 \pm 0.0}$

Results: unconditional model



Results: conditional model



Deep kernel infinite exponential models

A famous quote

"Combining a deep architecture with a kernel machine that takes the higher-level learned representation as input can be quite powerful."

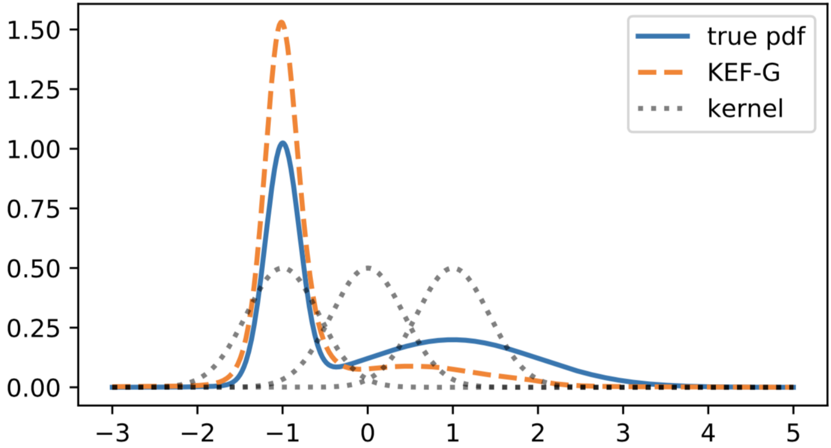
A famous quote

"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)

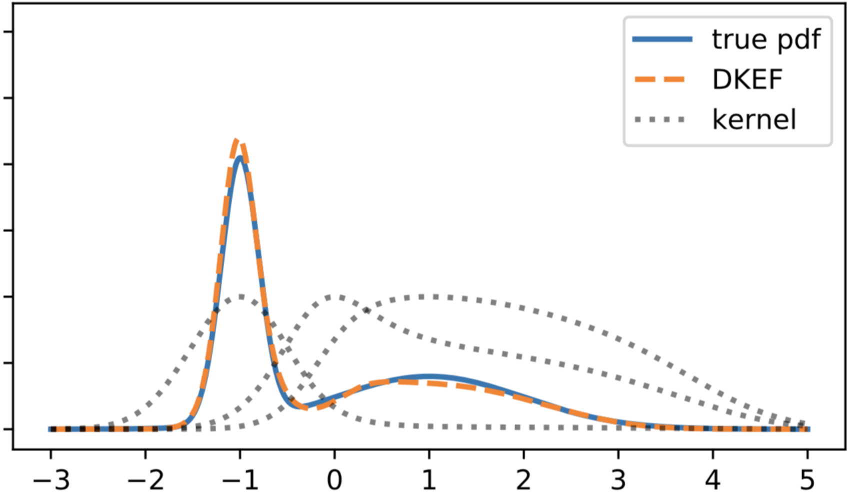
The case for nonstationary (learned) kernels

Stationary kernels, nonstationary target:



The case for nonstationary (learned) kernels

Nonstationary kernels, nonstationary target:



The model class

Nonstationary kernels, nonstationary target:

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

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

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

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

Define a "lite" model of the form:

$$f_{\boldsymbol{\alpha}, \boldsymbol{z}}^{k} := \sum_{m=1}^{M} \alpha_m k_{\boldsymbol{w}}(\boldsymbol{z}_m, \cdot)$$

where $\boldsymbol{w}$ are the kernel parameters (next slide).

Kernel design

Kernel of the form:

$$k_w(x, y) = \sum_{r=1}^{R} \rho_r \exp\left(-\frac{1}{2\sigma_r^2}\left\|\phi_{w_r}(x) - \phi_{w_r}(y)\right\|^2\right)$$

ϕ_{w_r} 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 $\tilde{p}^k_{\boldsymbol{\alpha},\boldsymbol{z}}$:

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

Comparison to earlier exponential family loss:

- The regulariser $\frac{\lambda_\alpha}{2}\|\boldsymbol{\alpha}\|^2$ is essential.
- Earlier work: primarily regularized with $\lambda_H \|f^k_{\boldsymbol{\alpha},\boldsymbol{z}}\|^2_{\mathcal{H}}$. As we change k, however, $\|f\|_{\mathcal{H}}$ changes meaning.
- The “curvature” term $\lambda_C \sum_{n=1}^{N}\sum_{d=1}^{D}\left[\partial_d^2 \log \tilde{p}^k_{\boldsymbol{\alpha},\boldsymbol{z}}(x_n)\right]^2$ 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 $\tilde{p}^k_{\boldsymbol{\alpha},\boldsymbol{z}}$:

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

Comparison to earlier exponential family loss:

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

The weights $\boldsymbol{\alpha}$ are solutions to a linear system

Nonstationary kernels, nonstationary target:

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

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

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

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

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

The algorithm

The challenge: we are optimising over many things:

- the locations of the inducing points, $\boldsymbol{z}$
- The parameters $\boldsymbol{w}$ of the convolutional features $\boldsymbol{\phi}$, including kernel weights $\boldsymbol{\rho}_r$.
- The regularisation coefficients $\boldsymbol{\lambda}_C$ and $\boldsymbol{\lambda}_\alpha$
- The coefficients $\boldsymbol{\alpha}$ themselves.

What doesn't work: joint optimisation over $\boldsymbol{w}, \alpha, \boldsymbol{\lambda}$. 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 ϕ, including kernel weights ρ_r.
- The regularisation coefficients λ_C and λ_α
- The coefficients α themselves.

What doesn't work: joint optimisation over w, α, λ. 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 α and λ.

Stage 1: learning $\boldsymbol{w}$ and $\boldsymbol{z}$

Fitting regulariser, inducing point:

While $\hat{J}(\tilde{p}^{k_{\boldsymbol{w}}}_{\boldsymbol{\alpha}(\boldsymbol{\lambda}, k_{\boldsymbol{w}}, \boldsymbol{z}, \mathcal{D}_1), \boldsymbol{z}}, \mathcal{D}_2)$ still improving do

- Sample disjoint data subsets $\mathcal{D}_t, \mathcal{D}_v \subset \mathcal{D}_1$
- Express natural parameter using inducing points, $f(\cdot) = \sum_{m=1}^{M} \alpha_m(\boldsymbol{\lambda}, k_{\boldsymbol{w}}, \boldsymbol{z}, \mathcal{D}_t) k_{\boldsymbol{w}}(\boldsymbol{z}_m, \cdot)$
    - α_m solved on training data $\mathcal{D}_t$.
- Define unregularised validation loss on $\mathcal{D}_v$:

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

- Take SGD steps in $\hat{J}$ for $\boldsymbol{w}$, $\boldsymbol{\lambda}$, and optionally $\boldsymbol{z}$.

Stage 2: refinement of $\boldsymbol{\lambda}$

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

While $\hat{J}(\tilde{p}^{k_w}_{\alpha(\boldsymbol{\lambda}, k_w, \boldsymbol{z}, \mathcal{D}_1), \boldsymbol{z}}, \mathcal{D}_2)$ still improving do

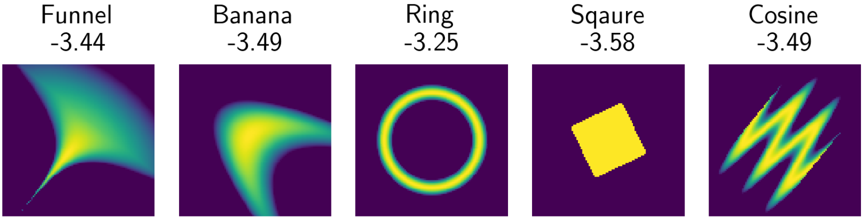
- Express natural parameter using inducing points, this time solving on all $\mathcal{D}_1$, $f(\cdot) = \sum_{m=1}^{M} \alpha_m(\boldsymbol{\lambda}, k_w, \boldsymbol{z}, \mathcal{D}_1) k_w(\boldsymbol{z}_m, \cdot)$
- Define unregularised validation loss on $\mathcal{D}_2$,

$$\hat{J} = \frac{1}{|\mathcal{D}_2|} \sum_{n=1}^{|\mathcal{D}_2|} \sum_{d=1}^{D} \left[\partial_d^2 f(x_n) + \frac{1}{2} (\partial_d f(x_n))^2 \right]$$

- Take SGD steps in $\hat{J}$ for $\boldsymbol{\lambda}$ only.

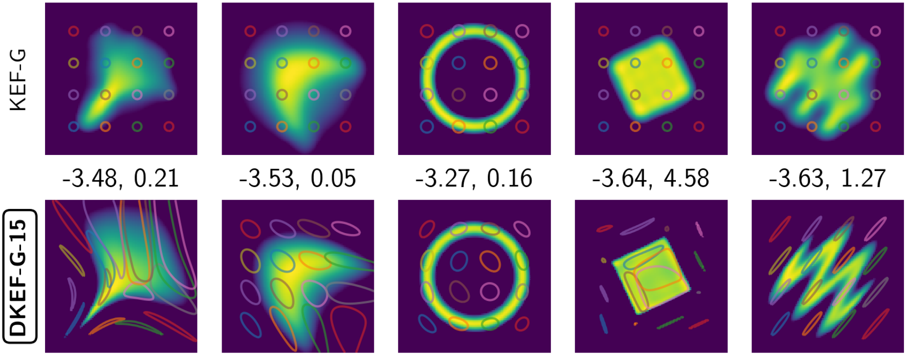
What works, what doesn't work, and why

"The usual suspects":



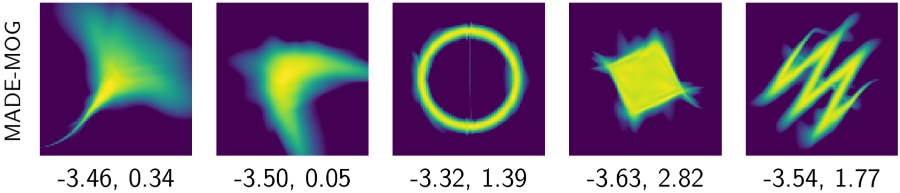
What works, what doesn't work, and why

Learned kernels vs fixed kernels:



What works, what doesn't work, and why

MADE with mixture of Gaussians:



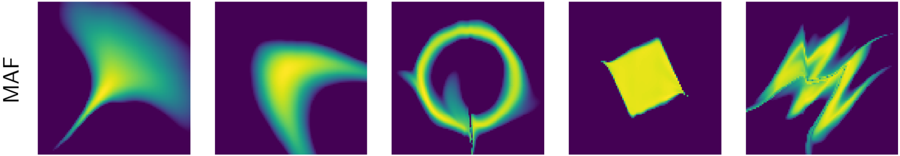
Definition of MADE (Masked Autoencoder for Distribution Estimation):

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

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)



Definition of masked autoregressive flow:

$$p(x_i|\mathbf{x}_{1:i-1}) = \mathcal{N}(x_i|\mu_i, (\exp\alpha_i)^2)$$

$$\mu_i = f_{\mu_i}(\mathbf{x}_{1:i-1})$$

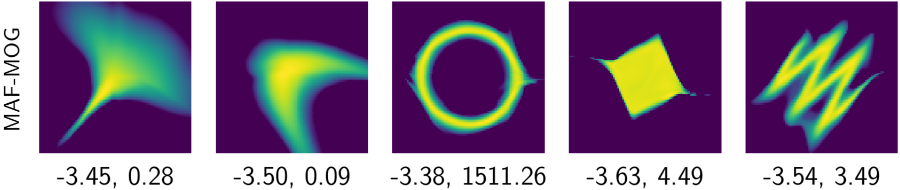
$$\alpha_i = f_{\alpha_i}(\mathbf{x}_{1:i-1})$$

$$x_i = u_i \exp(\alpha_i) + \mu_i$$

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



of masked autoregressive flow:

$$p(x_i | \mathbf{x}_{1:i-1}) = \mathcal{N}(x_i | \mu_i, (\exp \alpha_i)^2)$$

$$\mu_i = f_{\mu_i}(\mathbf{x}_{1:i-1})$$

$$\alpha_i = f_{\alpha_i}(\mathbf{x}_{1:i-1})$$

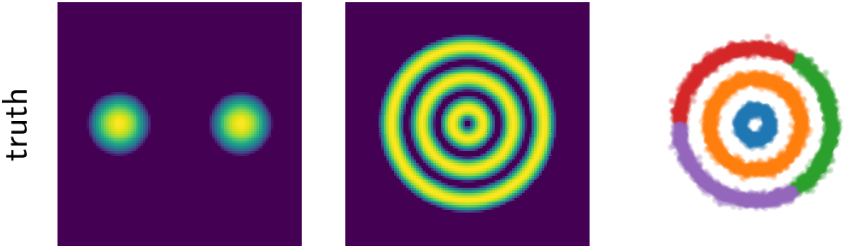
$$x_i = u_i \exp(\alpha_i) + \mu_i$$

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.

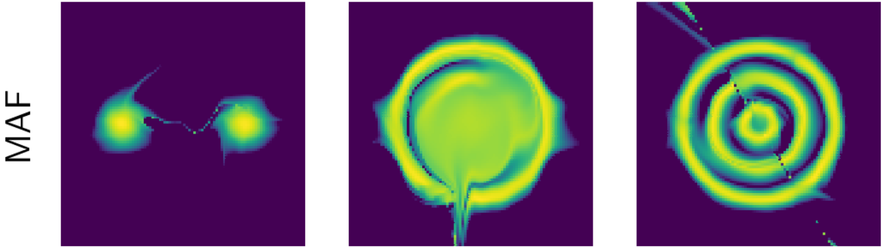
Two simple datasets

Disconnected mixture of two Gaussians, and bullseye:



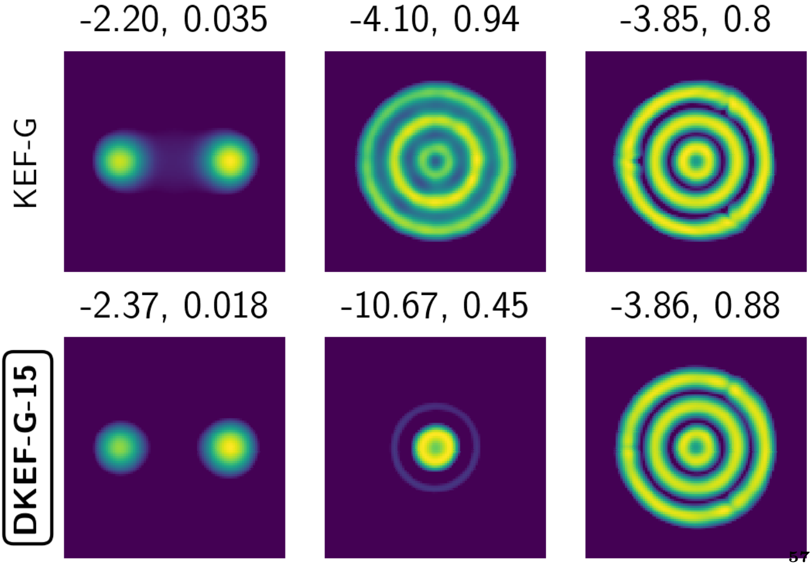
How does MAF do?

Disconnected mixture of two Gaussians, and bullseye:



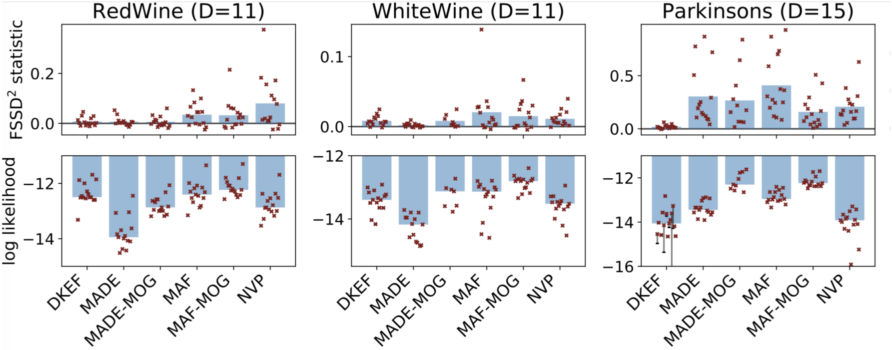
How does kernel exponential family do?

Disconnected mixture of two Gaussians, and bullseye:



Solutions, kernel Stein discrepancy and log likelihood

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



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 $\mathbf{f}$, labels $\mathbf{y}$, (with covariate matrix X), and hyperparameters θ:

$$p(\mathbf{f}, \mathbf{y}, \theta) = p(\theta)p(\mathbf{f}|\theta)p(\mathbf{y}|\mathbf{f})$$

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

- Automatic Relevance Determination (ARD) covariance:

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

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

$$p(y_i|f(x_i)) = (1 - \exp(-y_i f(x_i)))^{-1}, \qquad y_i \in \{-1, 1\}.$$

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 $\mathbf{f}$, labels $\mathbf{y}$, (with covariate matrix X), and hyperparameters θ:

$$p(\mathbf{f}, \mathbf{y}, \theta) = p(\theta)p(\mathbf{f}|\theta)p(\mathbf{y}|\mathbf{f})$$

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

- Automatic Relevance Determination (ARD) covariance:

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

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

$$p(y_i|f(x_i)) = (1 - \exp(-y_i f(x_i)))^{-1}, \qquad y_i \in \{-1, 1\}.$$

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 $\mathbf{f}$, labels $\mathbf{y}$, (with covariate matrix X), and hyperparameters θ:

$$p(\mathbf{f}, \mathbf{y}, \theta) = p(\theta)p(\mathbf{f}|\theta)p(\mathbf{y}|\mathbf{f})$$

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

- Automatic Relevance Determination (ARD) covariance:

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

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

$$p(y_i|f(x_i)) = (1 - \exp(-y_i f(x_i)))^{-1}, \qquad y_i \in \{-1, 1\}.$$

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 $\mathbf{f}$, labels $\mathbf{y}$, (with covariate matrix X), and hyperparameters θ:

$$p(\mathbf{f}, \mathbf{y}, \theta) = p(\theta)p(\mathbf{f}|\theta)p(\mathbf{y}|\mathbf{f})$$

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

- Automatic Relevance Determination (ARD) covariance:

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

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

$$p(y_i|f(x_i)) = (1 - \exp(-y_i f(x_i)))^{-1}, \qquad y_i \in \{-1, 1\}.$$

Bayesian Gaussian process classification

Example: when is target not computable?

- Gaussian process classification, latent process $\mathbf{f}$

$$p(\theta|\mathbf{y}) \propto p(\theta)p(\mathbf{y}|\theta) = p(\theta)\int p(\mathbf{f}|\theta)p(\mathbf{y}|\mathbf{f},\theta)\,d\mathbf{f} =: \pi(\theta)$$

... but cannot integrate out $\mathbf{f}$

- Metropolis Hastings ratio:

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

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

$$\hat{p}(\theta|\mathbf{y}) \propto p(\theta)\hat{p}(\mathbf{y}|\theta) \approx p(\theta)\frac{1}{n_{\text{imp}}}\sum_{i=1}^{n_{\text{imp}}} p(\mathbf{y}|\mathbf{f}^{(i)})\frac{p(\mathbf{f}^{(i)}|\theta)}{Q(\mathbf{f}^{(i)})}$$

Bayesian Gaussian process classification

Example: when is target not computable?

- Gaussian process classification, latent process $\mathbf{f}$

$$p(\theta|\mathbf{y}) \propto p(\theta)p(\mathbf{y}|\theta) = p(\theta)\int p(\mathbf{f}|\theta)p(\mathbf{y}|\mathbf{f},\theta)\,d\mathbf{f} =: \pi(\theta)$$

... but cannot integrate out $\mathbf{f}$

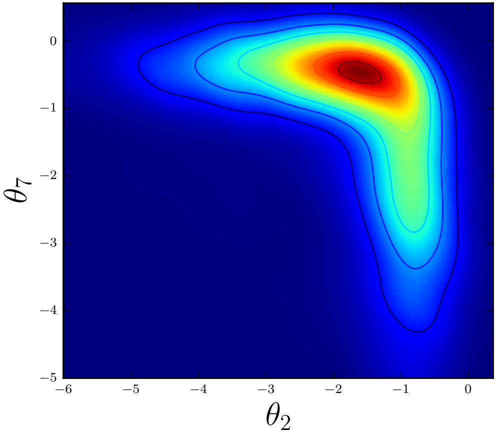
- Estimated MH ratio:

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

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

Adaptive HMC

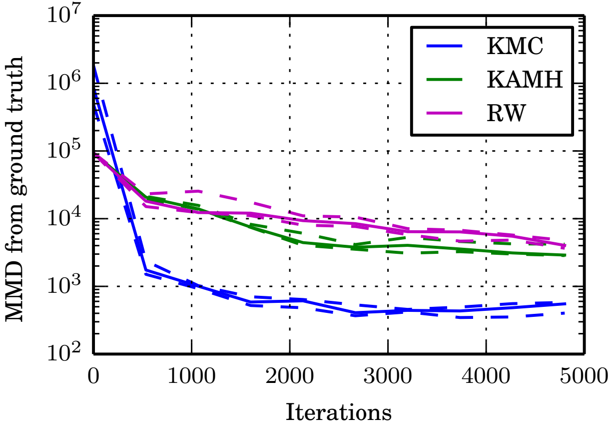
Sliced posterior over hyperparameters of a Gaussian Process classifier on UCI Glass dataset obtained using Pseudo-Marginal MCMC.



Can you learn an HMC sampler?

Basic adaptive Metropolis-Hastings

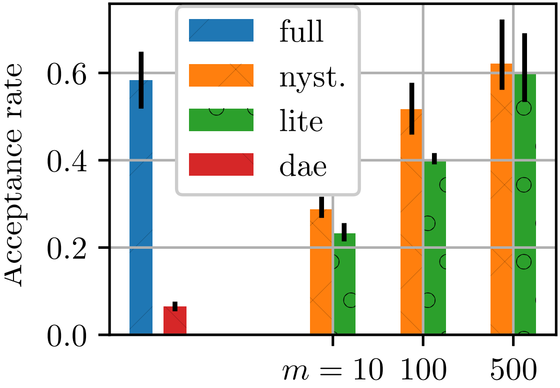
Sliced posterior over hyperparameters of a Gaussian Process classifier on UCI Glass dataset obtained using Pseudo-Marginal MCMC.



Significant improvements over random walk

Efficiency gains from approximate solution

HMC and aceptane rates for 90% quantiles



Co-authors

From Gatsby:

- Michael Arbel
- Kevin Li
- Heiko Strathmann
- Dougal Sutherland

External collaborators:

- Kenji Fukumizu
- Bharath Sriperumbudur

Questions?