Accelerating Statistical Inference with Machine Learning

Faster Statistical Computation through Flexible Approximation

Lingge Li

Department of Statistics
University of California Irvine

April 2018

Outline

1. Overview
2. Case studies
   - Neural Network Gradient Hamiltonian Monte Carlo
   - Estimating Neutrino Oscillation with Bayesian Optimization
3. Future work

Importance of computational statistics

> **On the contrary, I believe that Monte Carlo methods, in a limited sense, can be superior to mathematical analysis for certain kinds of applied inference. Using Monte Carlo methods frees the applied statistician to explore a great variety of models with relative ease, and thus statisticians can pursue the scientific goals of matching models to data more effectively and with less algebraic digression**

Figure: Donald Rubin on Monte Carlo methods in his 1984 paper *Bayesianly Justifiable and Relevant Frequency Calculations for the Applied Statistician*.

- Before MCMC, Bayesian analysis was heavily dependent on and limited to conjugate distributions
- Without numerical linear algebra (lm.fit), even linear regression would be difficult (calculating $(X^T X)^{-1}$)
- Better computational methods allow building more complex models on more data

Computational bottlenecks of statistical inference

- Big data
  - Computation burden $\mathcal{O}(f(n, d))$ scales with data where n is number of observations and d is dimensionality
  - Sampling/MCMC on large data sets is especially challenging
- Complex models
  - Models may not admit an explicit form for likelihood
  - Observations may come from a physical system described by differential equations

Borrow from machine learning

- Borrow from machine learning to address computational bottlenecks
- Neural network: scalable universal approximator
- Gaussian process: probabilistic kernel smoother
- Can approximate important but computationally expensive quantities

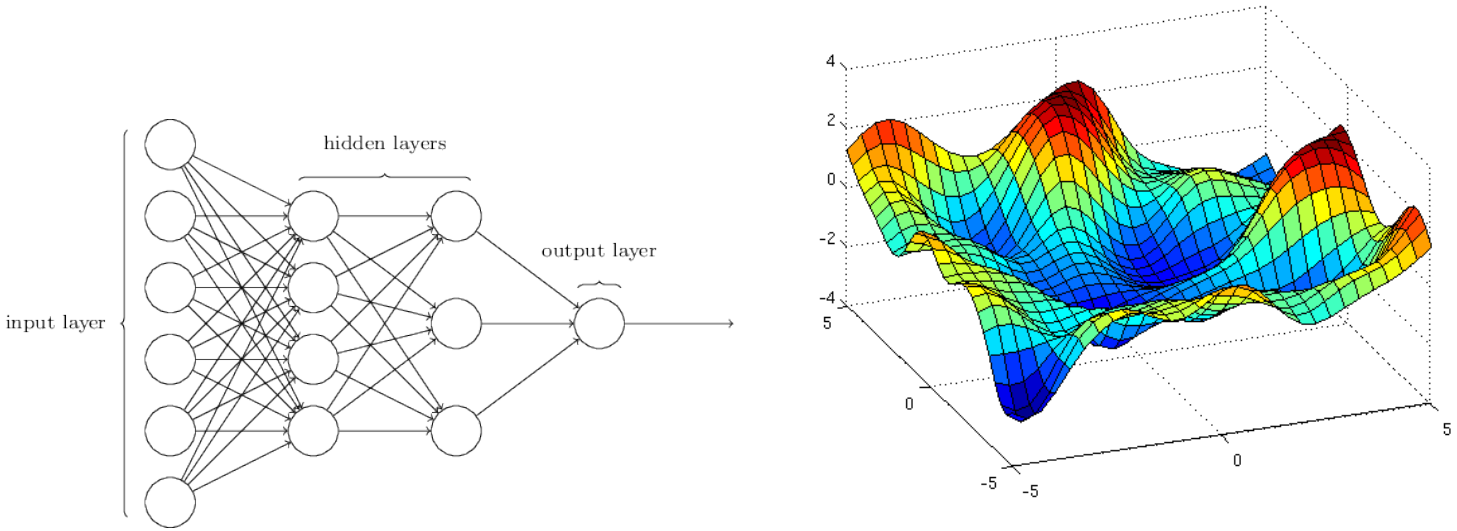

Figure: Feed forward neural network architecture (left) and realization of bivariate GP from David Duvenaud's thesis (right).

Gaussian process

- Gaussian process ($\mathcal{GP}$) is the extension of multivariate normal to infinite dimensions, a distribution in the function space
- $\mathcal{GP}(0, k(\cdot,\cdot))$ is characterized by kernel $k(x, x')$ that defines pairwise covariance

Let $f \sim \mathcal{GP}(0, k(\cdot,\cdot))$. Then

$$\begin{pmatrix} f(x) \\ f(x') \end{pmatrix} \sim \mathcal{N}(0, \begin{bmatrix} k(x,x) & k(x,x') \\ k(x,x') & k(x',x') \end{bmatrix}) \tag{1}$$

and

$$f(x')|f(x) \sim N(k(x,x')f(x)/k(x,x), k(x',x') - k(x,x')^2/k(x,x)). \tag{2}$$

Gaussian process sample paths

- Draw a sample path from $\mathcal{GP}(0, k(\cdot, \cdot))$ at $\vec{x}$
- Calculate covariance matrix $k(\vec{x}, \vec{x})$ and sample from $\mathcal{MVN}$
- Kernel k determines shape and smoothness

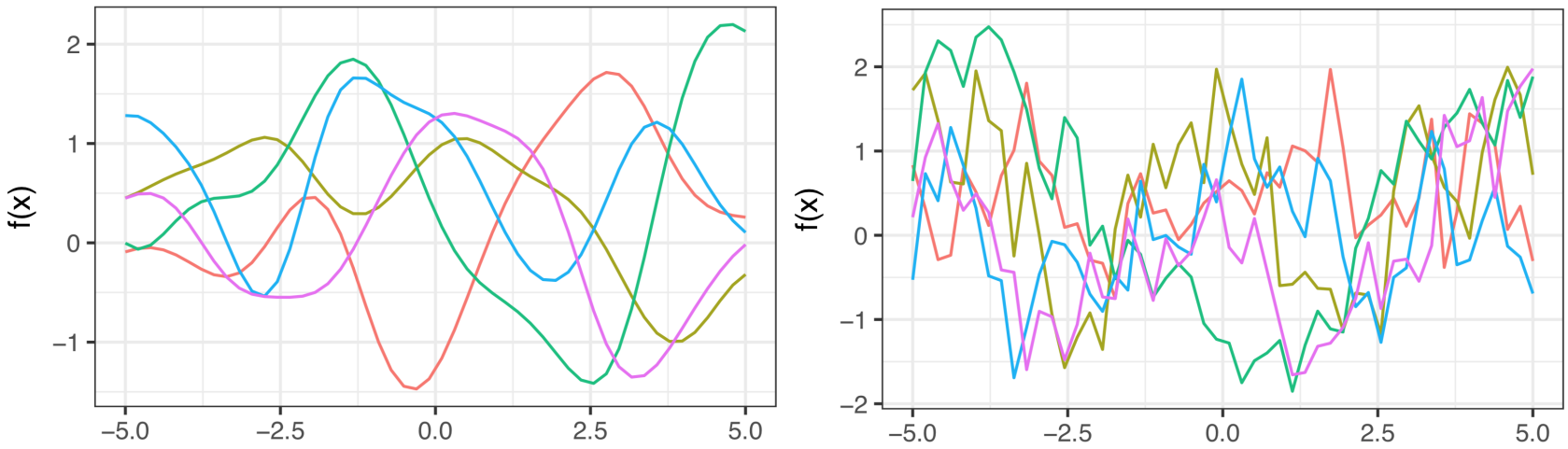

Figure: Squared exponential kernel $\exp(-(x - x')^2/2)$ vs Ornstein-Uhlenbeck kernel $\exp(-|x - x'|)$.

Gaussian process regression

- Given observed data and fixed kernel, $\mathcal{GP}$ posterior is the conditional distribution in (2)
- Highly certain at observed data points, less certain where observed data points are sparse
- Kernels are controlled by hyper-parameters θ, which can be learned from data

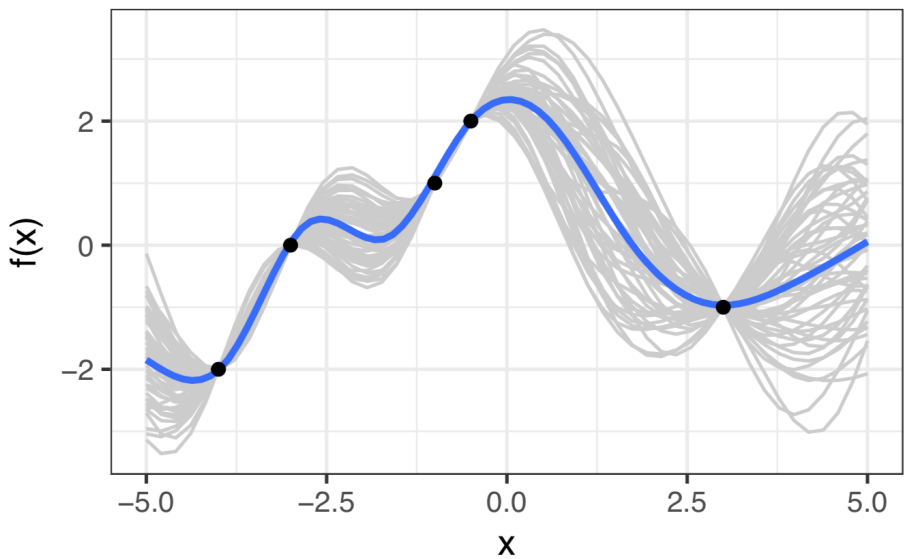

Figure: GP posterior with squared exponential kernel.

Neural network formulation

- Single neuron (perceptron) output is given by $g(\sum^i w_i x_i + b)$, where w_i is weight, b is bias, and g is activation
- Each neuron can be considered logistic regression and a neural network can be considered "stacked GLMs"
- Back-propagation for training to minimize loss function (gradient descent with chain rule)

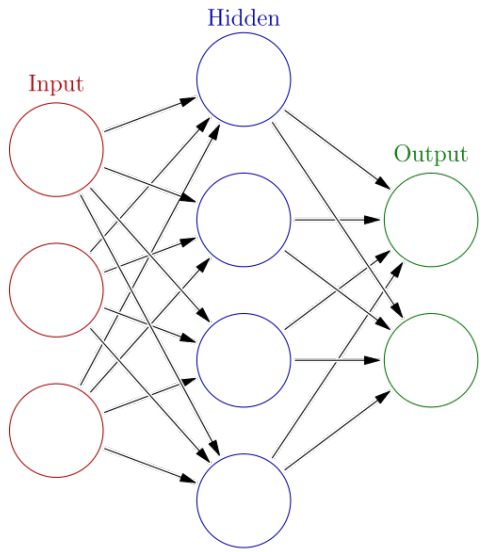

Figure: One hidden layer neural network.

Universal approximation theorem

- Existence of universal approximation is known (Cybenko 1989)

Theorem 3. *Let σ be a continuous sigmoidal function. Let f be the decision function for any finite measurable partition of I_n. For any $\varepsilon > 0$, there is a finite sum of the form*

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(y_j^T x + \theta_j)$$

and a set $D \subset I_n$, so that $m(D) \geq 1 - \varepsilon$ and

$$|G(x) - f(x)| < \varepsilon \qquad \text{for} \quad x \in D.$$

Metropolis-Hastings

$$x' \sim q(\cdot|x_t), A(x'|x_t) = \min(1, \frac{p(x')q(x_t|x')}{p(x_t)q(x'|x_t)})$$

- Proposal distribution q
- Metropolis-Hastings correction ensures detailed balance condition
- A good proposal distribution would utilize the information about the posterior distribution

Comparisons of MCMC algorithms

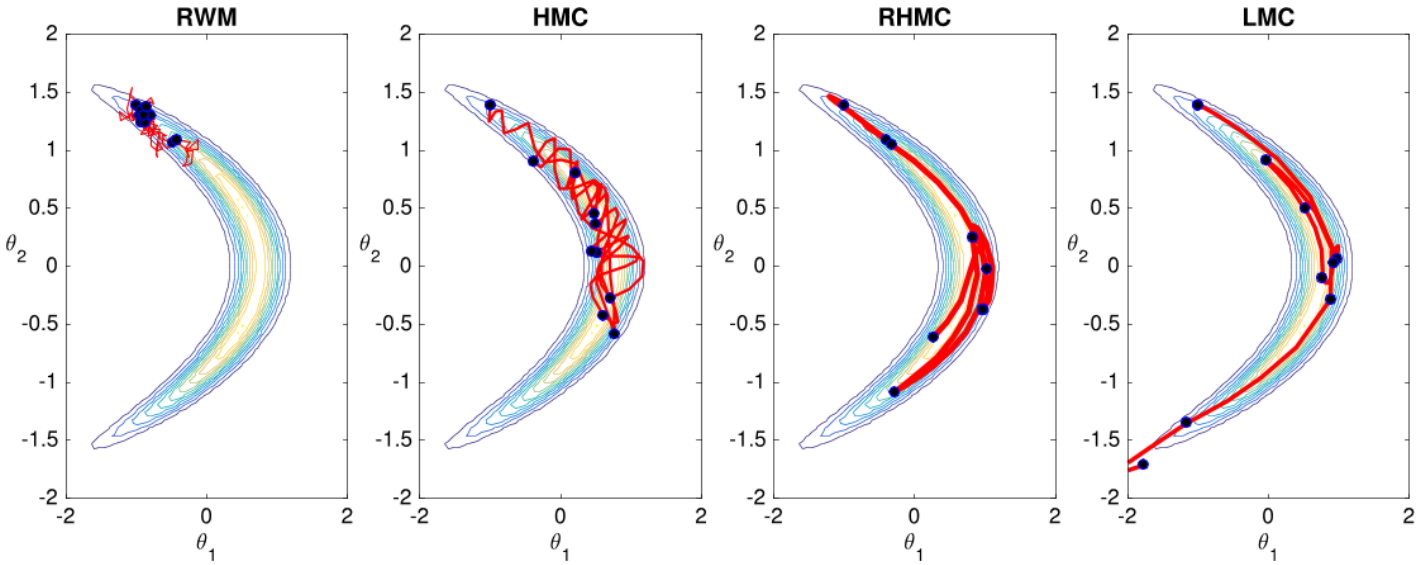

Figure: Random walk, Hamiltonian, Riemannian, and Langevin algorithms on the banana shaped distribution. Source: Shiwei Lan's website.

- Random walk does not work well as it "blindly" wanders
- More sophisticated algorithm such as Riemannian HMC (Girolami et al 2009) has higher computational complexity
- For a fixed computation budget, want an algorithm that yields large effective sample size (Gelman et al 2014)

Hamiltonian Monte Carlo

$$H(\theta, p) = U(\theta) + K(p) \tag{3}$$

$$U(\theta) = -\log p(X|\theta)p(\theta) \text{ log posterior} \tag{4}$$

$$K(p) = -\frac{1}{2}||p||^2 \text{ log Gaussian density} \tag{5}$$

- Hamiltonian system $H(\theta, p)$
  θ position, p momentum
- $U(\theta)$ potential energy, $K(p)$ kinetic energy
- Properties
  Time reversibility, preservation of volume, conservation of energy

Interpretation

- Bayesian skate park: momentum p is initial kick and $U(\theta)$ is gravitational energy
- Gain speed down-slope, gain potential up-slope (always accept)
- Can travel far with enough momentum (low auto-correlation)

Figure: Skatepark in Venice Beach.

Implementation

- Randomly sample momentum p from multivariate Gaussian
- Change θ, p according to Hamiltonian system $H(\theta, p)$

$$\frac{d\theta}{dt} = \frac{\partial H}{\partial p} = \frac{\partial K}{\partial p} \tag{6}$$

$$\frac{dp}{dt} = -\frac{\partial H}{\partial \theta} = -\frac{\partial U}{\partial \theta} \tag{7}$$

- Leapfrog numerical integration with L steps of size Δt (tuning parameters) to simulate trajectory
- MH correction to ensure convergence despite integration errors

Stochastic gradient HMC

- Repeated gradient calculation can be expensive as it needs all the data
- Sub-sample data and calculate gradient instead
- Stochastic Langevin dynamics (one step HMC) by Welling et al 2011
- Add "friction" term to counterbalance noisy gradient by Chen, Fox, and Guestrin 2014

$$\frac{dp}{dt} = -\nabla U(\theta) + \mathcal{N}(0, 2B) - BM^{-1}p \tag{8}$$

Problem with stochastic gradient

- Noise "multiplies" rather than "averages out" during leapfrog integration
- With MH correction, the acceptance probability drops substantially
- "Friction" term is *ad hoc*; more like SGD with momemtum
- Without MH correction, the sampler roughly converges to the correct location but misses the shape of distribution

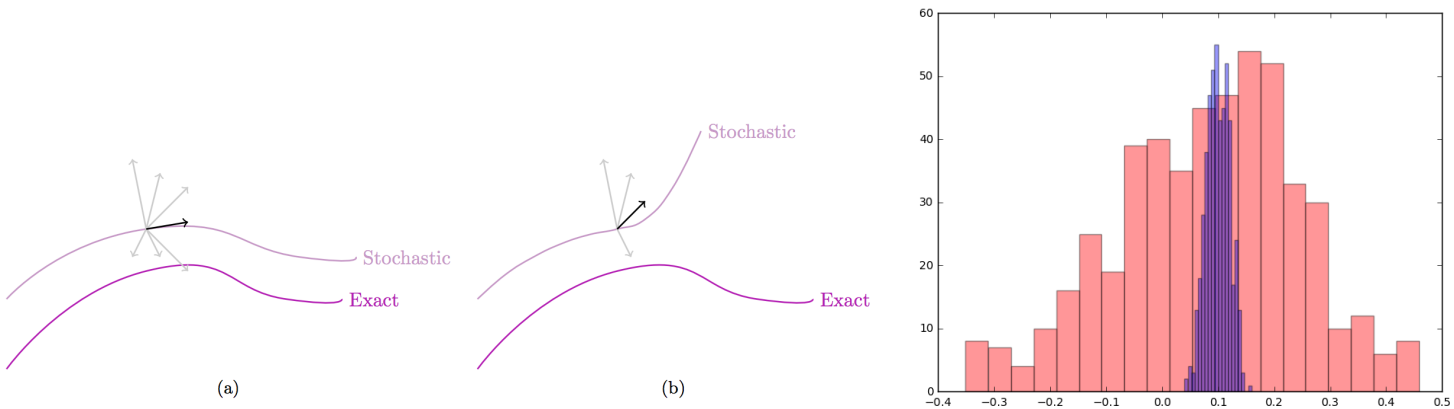

Figure: Comparison of stochastic and exact leapfrog trajectories from Betancourt 2017 (left). SGHMC posterior (blue) vs standard HMC posterior (red) on UCI News data.

Surrogate HMC

- Use a surrogate function to approximate U and use $\nabla \hat{U}$
- Collect training data for the surrogate function during burn-in
- Gaussian process surrogate by Rasmussen 2008
  GP surrogate itself is computationally expensive and does not scale
- Shallow neural network surrogate by Zhang, Shahbaba, and Zhao 2017
  Hidden layer weights are randomly initialized and fixed (extreme learning machine)

Our approach

- Directly model ∇U with a neural network and use $\widehat{\nabla U}$
  The gradient field is the most important part of HMC
- Still single hidden layer but train with back-propagation
  Though random projection is powerful, trained hidden layers yield more accurate approximation with fewer units
- More training data and information
  Can have surrogate S close to energy U but ∇S different from ∇U

Gaussian example

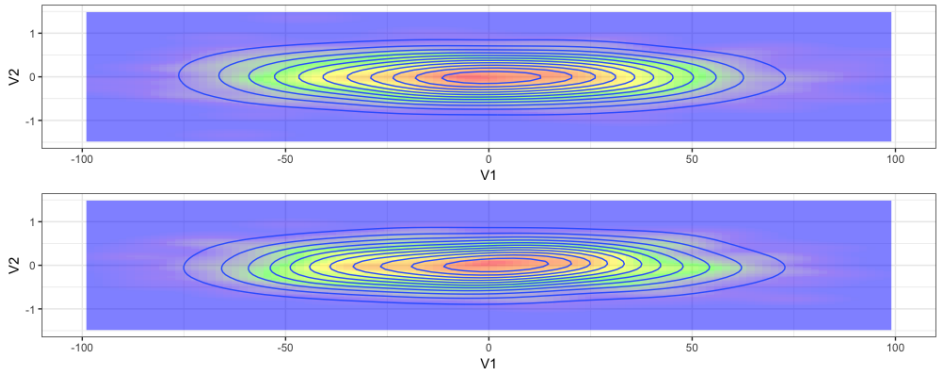

Figure: NNGHMC and standard HMC posteriors of ill-conditioned multivariate Gaussian.

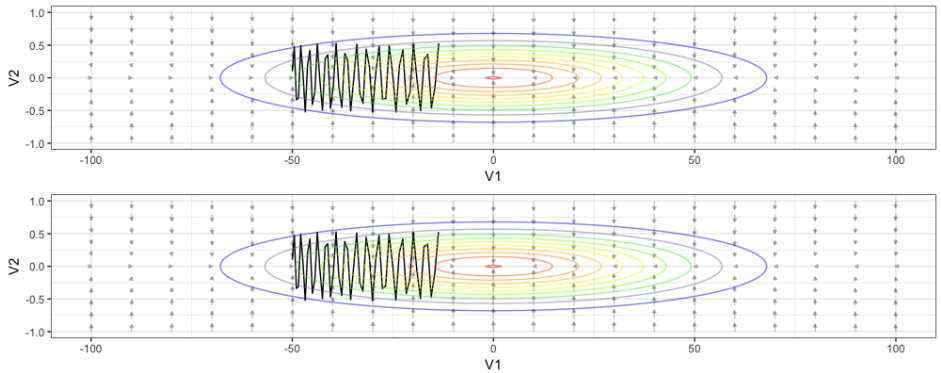

Figure: Leapfrog trajectories using NNGHMC and standard HMC on the same distribution.

Summary

Table 1: Acceptance probability when sampling from multivariate Gaussian

Method	Dimension / Training	500	1000	2000
Gaussian process	10	0.65	0.61	0.57
	20	0.64	0.65	0.62
	40	0.31	0.32	0.32
Neural network gradient	10	0.95	0.96	0.97
	20	0.82	0.87	0.91
	40	0.61	0.75	0.87

Table 2: Experiment results on News data

Method	AP	ESS	CPU time	Median ESS/s	Speed-up
Standard	0.77	(777, 2021, 5929)	3607s	0.66	1
NNg (10%)	0.61	(605, 1416, 4865)	502s	2.82	4.27
NNg (15%)	0.64	(620, 1382, 5500)	678s	2.04	3.09
NNg (20%)	0.68	(700, 1731, 5397)	854s	2.03	3.08

AP: acceptance probability
ESS: effective sample size (min, median, max) after removing 10% burn-in

- More scalable than GP surrogate
- Slightly lower acceptance probability but higher ESS given computation budget compared to standard HMC

Training schedule

- How much training data is enough and how long to train neural network for?
- Training schedule under computation budget
- Collect true gradient with standard HMC, train neural network, run HMC with $\widehat{\nabla U}$, and repeat...

Neutrino oscillation experiment

$$P(\nu_\mu \to \nu_e) = f(Energy; \sin^2\theta_{23}, \Delta m^2_{32}, \delta_{CP}) \tag{9}$$

- Send a beam of muon neutrinos into the particle detector
- Muon neutrinos oscillate to electron neutrinos according to (9)
- Observe oscillated electron neutrinos and measure their energy

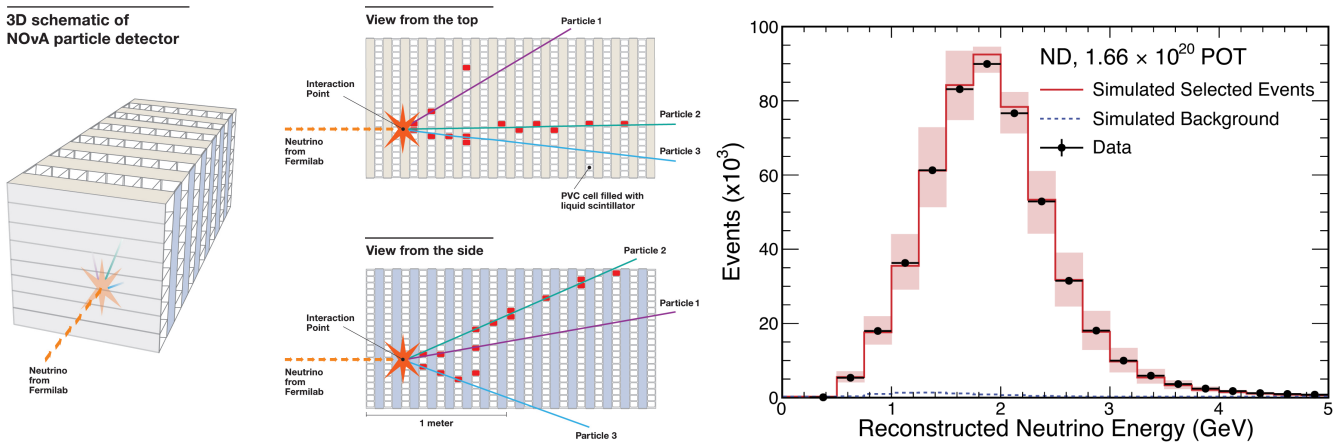

Figure: NOvA detector (left) and energy spectrum (right) from Adamson et al 2016.

Statistical model

- Observed data are neutrino energy $E_1, E_2, ..., E_n$; oscillation paramters θ in (9) and nuisance parameters (systematic errors) δ
- Using fixed k bins, the observed data have counts $X_1, X_2, ..., X_k$ and we expect counts $\lambda_1, \lambda_2, ..., \lambda_k$ given θ, δ
- There is an implicit mapping v between θ, δ and $\{\lambda_k\}$

$$X_k \sim Poisson(\lambda_k) \text{ where } \lambda_k = v(\theta, \delta)_k \tag{10}$$

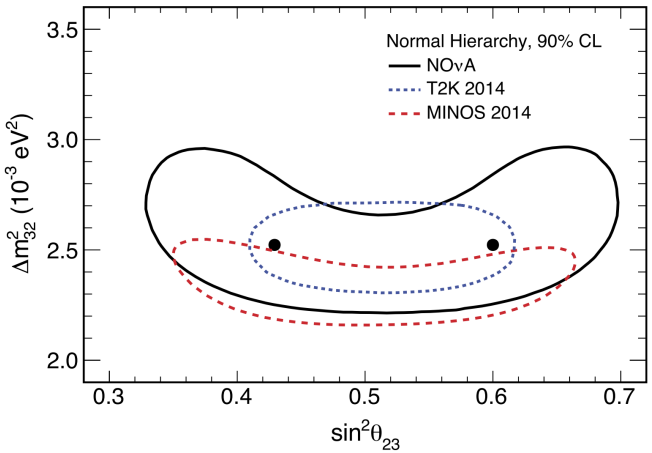

Figure: The inferential goal is a contour (Adamson et al 2016).

Feldman-Cousins method

- Inverted likelihood ratio test (profile likelihood)
  For fixed oscillation parameters θ_0

$$\Lambda = \frac{L(\theta_0, \hat{\hat{\delta}})}{L(\hat{\theta}, \hat{\delta})} \text{ where } \hat{\hat{\delta}} \text{ is the best fit given } \theta_0. \tag{11}$$

- Most powerful by Neyman-Pearson lemma
- Asymptotic χ^2 distribution (Wilks' theorem) does not hold as we only have one observation from each of the k Poisson distributions
- Simulate reference distribution instead with $\delta = \hat{\delta}_X$ fixed at best fit to actual data (Feldman and Cousins 1998)

Feldman-Cousins method details

- Loop through θ_i on a grid of size N in bounded parameter space
- Generate M sets of data $X^{j=1,\ldots,M}$ with $v(\theta_i, \hat{\delta}_X)$ following (10)
- Fit parameters $\hat{\hat{\delta}}, \hat{\theta}, \hat{\delta}$ in (11) to each X^j and calculate $-2\log\Lambda$

As fitting involves optimizing implicit function v and the whole process has NM iterations, it is very expensive computationally.

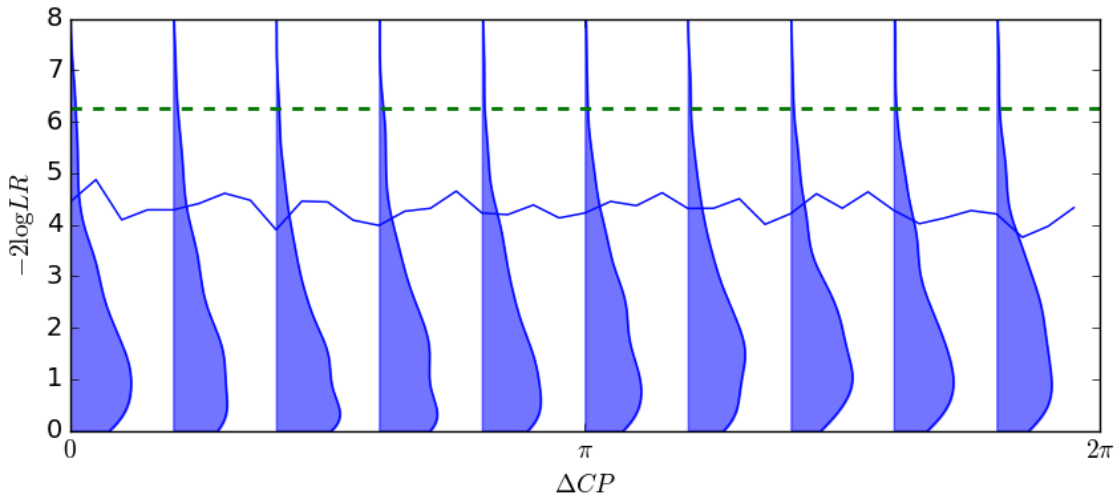

Figure: Asymptotic χ^2 distribution can be conservative, therefore creating a bigger contour.

Bayesian optimization

- Function f to be optimized that is expensive to evaluate
- Gaussian process as a probabilistic model for f
- Acquisition function balances exploration and exploitation
- Very popular method for tuning ML model hyper-parameters
- Optimal experimental design with a sequence of designs and polynomial model (Box and Wilson 1951)

Bayesian optimization

- Train Gaussian process on observed data points $\vec{x}_{obs}, f(\vec{x}_{obs})$
- With GP posterior, we can calculate probability of improvement, expectation of improvement, and uncertainty bound at new point x^*
- Evaluate f at the most promising point and repeat

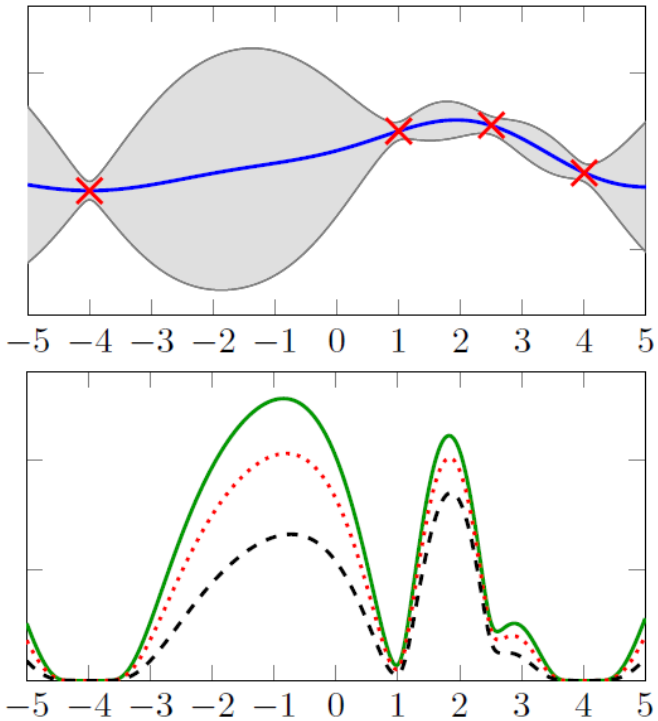

Figure: Canonical example from Snoek, Larochelle, and Adams 2012.

Adaptive search

- Gaussian process to model LRT thresholds

$$f(\theta) = T_\theta = [-2 \log \Lambda(\theta)]_{(1-\alpha)}$$

- Acquisition function targets points in the parameter space (of θ) that are unsure whether to accept/reject

$$a(\theta) = |\frac{\hat{T}(\theta) - [-2 \log \Lambda(\theta)^{|X}]}{\sigma_{\hat{T}(\theta)}}|^{-1}$$

- Save computation (reduce N) in the parameter space where LRT would definitely accept/reject

Adaptive search

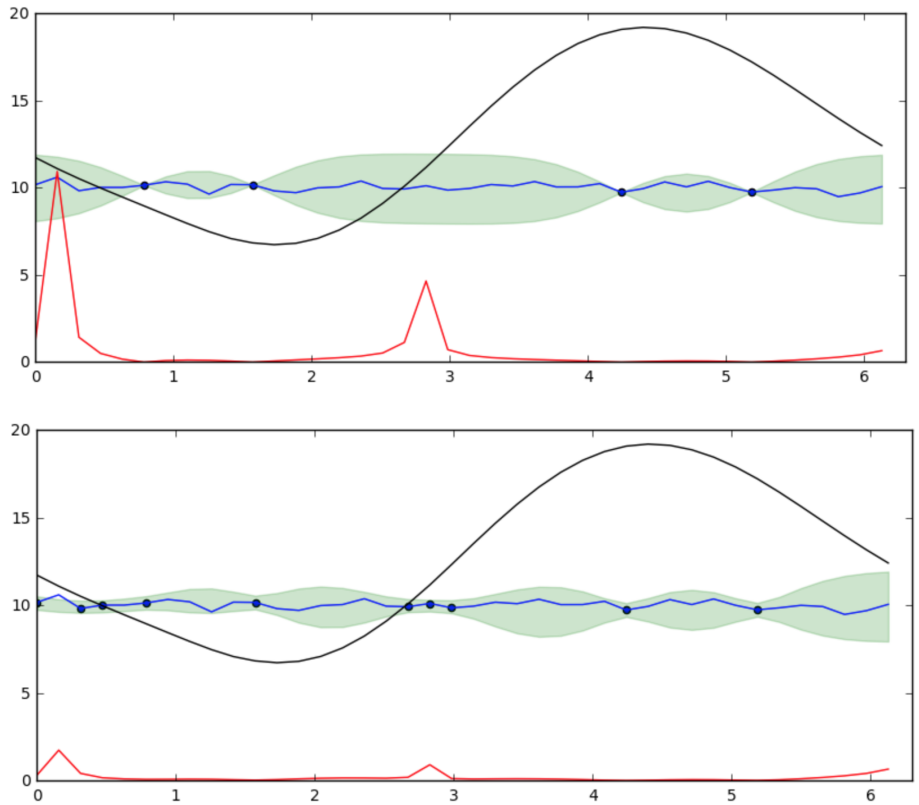

Figure: 1-dimensional illustrative example where the black curve is the observed likelihood, the blue curve is the LRT threshold, the green band is the GP surrogate, and the red curve is the acquisition function.

Considerations

- Choice of GP kernel and hyper-parameters priors
  Is there a kernel more appropriate than squared exponential? How to avoid over-fitting and over-smoothing?
- Different acquisition functions
  Current acquisition function may be overly greedy?
- Multiple approximations/coverage
  Can we jointly control multiple levels of approximation?

Fully Bayesian alternative

- Put weakly informative priors on θ and draw from posterior with HMC
- Gradient intractable analytically because of implicit mapping v between θ, δ and $\{\lambda_k\}$
- Could use numerical gradient as often done in optimization but it is computationally expensive
- Surrogate HMC?! (Lan et al 2015)

Model extensions

- Poisson likelihood comes from histogram density estimation with arbitrary binning
  Cox-process: Poisson point process with Gaussian process mean
- There are way more nuisance parameters δ than oscillation parameters θ
  Properly model nuisance parameters δ and their correlation δ
- These model extensions would not be possible without faster computational methods

Next steps

- Apply neural network gradient HMC on a variety of models such as stochastic volatility models
- Explore possible trade-offs of deeper/recurrent neural networks
- Implement the adaptive method for real neutrino data analysis and generalize to similar experiments
- Use approximation to balance "exact" statistical inference against computational bottlenecks

References

1. Rubin, Donald B. "Bayesianly justifiable and relevant frequency calculations for the applied statistician."The Annals of Statistics12.4 (1984): 1151-1172.
2. Duvenaud, David.Automatic model construction with Gaussian processes. Diss. University of Cambridge, 2014.
3. Williams, Christopher KI, and Carl Edward Rasmussen. "Gaussian processes for machine learning."the MIT Press2.3 (2006): 4.
4. Cybenko, George. "Approximation by superpositions of a sigmoidal function."Mathematics of control, signals and systems2.4 (1989): 303-314.

References

5. Girolami, Mark, Ben Calderhead, and Siu A. Chin. "Riemannian manifold hamiltonian monte carlo."arXiv preprint arXiv:0907.1100(2009).
6. Gelman, Andrew, et al.Bayesian data analysis. Vol. 2. Boca Raton, FL: CRC press, 2014.
7. Neal, Radford M. "MCMC using Hamiltonian dynamics."Handbook of Markov Chain Monte Carlo2.11 (2011).
8. Betancourt, Michael. "A conceptual introduction to Hamiltonian Monte Carlo."arXiv preprint arXiv:1701.02434(2017).

References

9. Welling, Max, and Yee W. Teh. "Bayesian learning via stochastic gradient Langevin dynamics."Proceedings of the 28th International Conference on Machine Learning (ICML-11). 2011.
10. Chen, Tianqi, Emily Fox, and Carlos Guestrin. "Stochastic gradient hamiltonian monte carlo."International Conference on Machine Learning. 2014.
11. Bernardo, J., et al. "Gaussian processes to speed up hybrid Monte Carlo for expensive Bayesian integrals."Bayesian statistics7 (2008): 651-659.
12. Zhang, Cheng, Babak Shahbaba, and Hongkai Zhao. "Hamiltonian Monte Carlo acceleration using surrogate functions with random bases."Statistics and computing27.6 (2017): 1473-1490.

References

13. Adamson, P., et al. "First measurement of muon-neutrino disappearance in NOvA."Physical Review D93.5 (2016): 051104.
14. Feldman, Gary J., and Robert D. Cousins. "Unified approach to the classical statistical analysis of small signals."Physical Review D57.7 (1998): 3873.
15. Snoek, Jasper, Hugo Larochelle, and Ryan P. Adams. "Practical bayesian optimization of machine learning algorithms."Advances in neural information processing systems. 2012.
16. Box, George EP, and Kenneth B. Wilson. "On the experimental attainment of optimum conditions."Breakthroughs in statistics. Springer, New York, NY, 1992. 270-310.
17. Lan, Shiwei, et al. "Emulation of higher-order tensors in manifold Monte Carlo methods for Bayesian Inverse Problems."Journal of Computational Physics308 (2016): 81-101.

Gaussian process

- Kernels

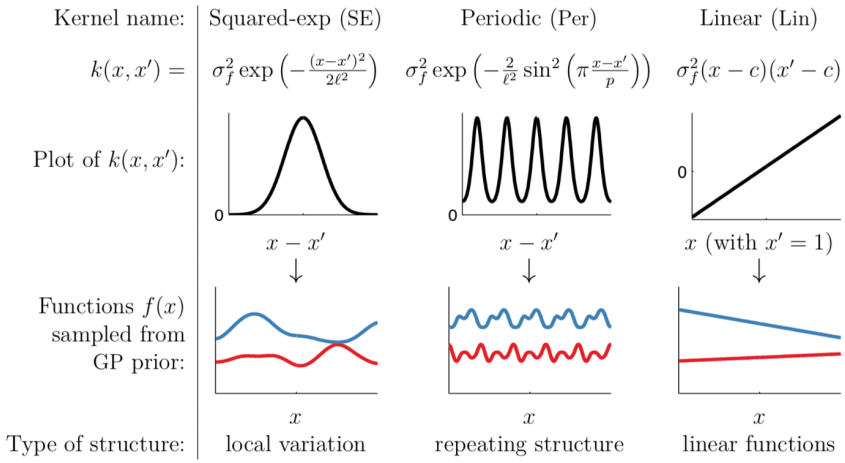

Figure 1.1: Examples of structures expressible by some basic kernels.

- Hyper-parameters
  l length-scale, σ_f^2 variance in squared-exponential kernel
- Training

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X)+\sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right). \tag{2.21}$$

Deriving the conditional distribution corresponding to eq. (2.19) we arrive at the key predictive equations for Gaussian process regression

$$\mathbf{f}_*|X,\mathbf{y},X_* \sim \mathcal{N}(\bar{\mathbf{f}}_*,\ \mathrm{cov}(\mathbf{f}_*)), \quad \text{where} \tag{2.22}$$

$$\bar{\mathbf{f}}_* \triangleq \mathbb{E}[\mathbf{f}_*|X,\mathbf{y},X_*] = K(X_*,X)[K(X,X)+\sigma_n^2 I]^{-1}\mathbf{y}, \tag{2.23}$$

$$\mathrm{cov}(\mathbf{f}_*) = K(X_*,X_*) - K(X_*,X)[K(X,X)+\sigma_n^2 I]^{-1}K(X,X_*). \tag{2.24}$$

MCMC theory

- Detailed balance condition (reversible)

$$p(x')q(x_t|x') = q(x'|x_t)p(x_t)$$

- Proposal distribution q (ergodic: aperiodic and positive recurrent)

this convergence to the invariant distribution occurs under mild regularity conditions. The regularity conditions required are irreducibility and aperiodicity [see Smith and Roberts (1993)]. What these mean is that, if x and y are in the domain of $\pi(\cdot)$, it must be possible to move from x to dy in a finite number of iterations with nonzero probability, and the number of moves required to move from x to dy is not required to be a multiple of some integer. These conditions are usually satisfied if $q(x, y)$ has a positive density on the same support as that of $\pi(\cdot)$. It is usually also satis-

Figure: Chib and Greenberg 1995

HMC

- Time reversibility due to leapfrog

The leapfrog method. Even better results can be obtained with the *leapfrog* method, which works as follows:

$$p_i(t+\varepsilon/2) = p_i(t) - (\varepsilon/2)\frac{\partial U}{\partial q_i}(q(t)) \tag{2.28}$$

$$q_i(t+\varepsilon) = q_i(t) + \varepsilon\frac{p_i(t+\varepsilon/2)}{m_i} \tag{2.29}$$

$$p_i(t+\varepsilon) = p_i(t+\varepsilon/2) - (\varepsilon/2)\frac{\partial U}{\partial q_i}(q(t+\varepsilon)) \tag{2.30}$$

- Conservation of Hamiltonian, preservation of volume (density depends on height and volume) ensure detailed balance condition
- Approximated gradient field (shallow neural network) has zero divergence (Liouvilles Theorem)

Bayesian optimization

- Assumption of kernel

The power of the Gaussian process to express a rich distribution on functions rests solely on the shoulders of the covariance function. While non-degenerate covariance functions correspond to infinite bases, they nevertheless can correspond to strong assumptions regarding likely functions. In particular, the automatic relevance determination (ARD) *squared exponential* kernel

$$K_{\mathsf{SE}}(\mathbf{x}, \mathbf{x}') = \theta_0 \exp\left\{-\frac{1}{2} r^2(\mathbf{x}, \mathbf{x}')\right\} \qquad r^2(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^{D} (x_d - x'_d)^2 / \theta_d^2. \tag{4}$$

is often a default choice for Gaussian process regression. However, sample functions with this covariance function are unrealistically smooth for practical optimization problems. We instead propose the use of the ARD Matérn $5/2$ kernel:

$$K_{\mathsf{M52}}(\mathbf{x}, \mathbf{x}') = \theta_0 \left(1 + \sqrt{5 r^2(\mathbf{x}, \mathbf{x}')} + \frac{5}{3} r^2(\mathbf{x}, \mathbf{x}')\right) \exp\left\{-\sqrt{5 r^2(\mathbf{x}, \mathbf{x}')}\right\}. \tag{5}$$

- Acquisition function (integrate over hyper-parameters)

However, for a fully-Bayesian treatment of hyperparameters (summarized here by θ alone), it is desirable to marginalize over hyperparameters and compute the *integrated acquisition function*:

$$\hat{a}(\mathbf{x}\,;\, \{\mathbf{x}_n, y_n\}) = \int a(\mathbf{x}\,;\, \{\mathbf{x}_n, y_n\}, \theta)\, p(\theta \,|\, \{\mathbf{x}_n, y_n\}_{n=1}^{N})\, \mathrm{d}\theta, \tag{6}$$