Introduction to Gaussian Processes

Maurizio Filippone

EURECOM, Sophia Antipolis, France

June 19th, 2019
Suggested readings

Gaussian Processes for Machine Learning
Carl E. Rasmussen and Christopher K. I. Williams

Pattern Recognition and Machine Learning
C. Bishop
1. Probabilistic Reasoning

2. Bayesian Linear Models

3. Gaussian Processes
   - Bayesian Linear Model with infinite Basis Functions
   - Connections with Deep Neural Nets
   - Optimizing Kernel Parameters
   - Gaussian Processes beyond Regression

4. Scalable Gaussian Processes
Probabilistic Reasoning
Motivation

- Climate modeling

Motivation

- Earthquake modeling

Motivation

- Classification of neurodegenerative diseases

Filippone et al., AoAS, 2012
Motivation

- Coal mining disaster data

Hensman, Matthews, Filippone, Ghahramani, *NIPS*, 2015
A Unified Framework

A model might be expensive to simulate/inaccurate

- Emulate model/discrepancy using a surrogate
A Unified Framework

A model might be expensive to simulate/inaccurate

- Emulate model/discrepancy using a surrogate

A model might not even be available

- Replace it with a flexible model
A Unified Framework

A model might be expensive to simulate/inaccurate

- Emulate model/discrepancy using a surrogate

A model might not even be available

- Replace it with a flexible model

Gaussian Process Models for Accurate Quantification of Uncertainty
• Decision Theory = Probabilistic reasoning + Utility theory
• Decision Theory = Probabilistic reasoning + Utility theory

• Data viewed as random variables

• Probabilities as degrees of belief
Consider two continuous random variables $x$ and $y$

- **Sum rule:**
  \[ p(x) = \int p(x, y) \, dy \]

- **Product rule:**
  \[ p(x, y) = p(x|y) p(y) = p(y|x) p(x) \]
Consider two continuous random variables $x$ and $y$

- **Sum rule:**
  \[ p(x) = \int p(x, y) \, dy \]

- **Product rule:**
  \[ p(x, y) = p(x|y)p(y) = p(y|x)p(x) \]

- **Bayes’ rule:**
  \[ p(y|x) = \frac{p(x|y)p(y)}{p(x)} \]

- **NOTE**: Bayes’ rule is a direct consequence of the product rule
• Take these two examples

![Graph 1](image1.png)

![Graph 2](image2.png)

• We are interested in estimating a function $f(x)$ from data
• Most problems in Machine Learning can be cast this way!
• Implement a linear combination of basis functions

\[ f(x) = w^\top \varphi(x) \]

with

\[ \varphi(x) = (\varphi_1(x), \ldots, \varphi_D(x))^\top \]
Probabilistic Interpretation of Loss Minimization

- **Inputs**: $X = (x_1, \ldots, x_N)^T$
- **Labels**: $y = (y_1, \ldots, y_N)^T$
- **Weights**: $w = (w_1, \ldots, w_D)^T$

Quadratic Loss

$p(y|X, w) \propto \exp(-\text{Loss})$

- Minimization of a loss function
- ... equivalent as maximizing likelihood $p(y|X, w)$
Bayesian Inference

- **Inputs**: $X = (x_1, \ldots, x_N)^T$
- **Labels**: $y = (y_1, \ldots, y_N)^T$
- **Weights**: $w = (w_1, \ldots, w_D)^T$

$$p(w) \quad p(w|y, X)$$

$$p(w|y, X) = \frac{p(y|X, w)p(w)}{\int p(y|X, w)p(w)dw}$$
What do Bayesian Linear Models Have to Offer?

- Regression example
What do Bayesian Linear Models Have to Offer?

- Classification example
Bayesian Linear Models
Bayesian Linear Regression

• Modeling observations as noisy realizations of a linear combination of the features:

\[ p(y|w, X, \sigma^2) = \mathcal{N}(Xw, \sigma^2 I) \]

• Note: we dropped basis functions for now, so \( \varphi(x) = x \).
Bayesian Linear Regression

- Modeling observations as noisy realizations of a linear combination of the features:

\[ p(y|w, X, \sigma^2) = \mathcal{N}(Xw, \sigma^2 I) \]

- Note: we dropped basis functions for now, so \( \varphi(x) = x \).

- Gaussian prior over model parameters:

\[ p(w) = \mathcal{N}(0, S) \]
Bayesian Linear Regression

- Bayes rule:

\[
p(w|X, y) = \frac{p(y|X, w)p(w)}{\int p(y|X, w)p(w)dw} = \frac{p(y|X, w)p(w)}{p(y|X)}
\]
Bayesian Linear Regression

- **Bayes rule:**

  \[
  p(w|X, y) = \frac{p(y|X, w)p(w)}{\int p(y|X, w)p(w)dw} = \frac{p(y|X, w)p(w)}{p(y|X)}
  \]

- **Posterior density:** \( p(w|X, y) \)
  - Distribution over parameters *after* observing data
Bayesian Linear Regression

- **Bayes rule:**
  \[ p(w|X, y) = \frac{p(y|X, w)p(w)}{\int p(y|X, w)p(w)dw} = \frac{p(y|X, w)p(w)}{p(y|X)} \]

- **Posterior density:** \( p(w|X, y) \)
  - Distribution over parameters *after* observing data

- **Likelihood:** \( p(y|X, w) \)
  - Measure of “fitness”
Bayesian Linear Regression

- **Bayes rule:**
  \[
p(w|X, y) = \frac{p(y|X, w)p(w)}{\int p(y|X, w)p(w)dw} = \frac{p(y|X, w)p(w)}{p(y|X)}
\]

- **Posterior density:** \( p(w|X, y) \)
  - Distribution over parameters after observing data

- **Likelihood:** \( p(y|X, w) \)
  - Measure of “fitness”

- **Prior density:** \( p(w) \)
  - Anything we know about parameters before we see any data
Bayesian Linear Regression

- Bayes rule:

\[
p(w|X, y) = \frac{p(y|X, w)p(w)}{\int p(y|X, w)p(w)dw} = \frac{p(y|X, w)p(w)}{p(y|X)}
\]

- **Posterior density**: \( p(w|X, y) \)
  - Distribution over parameters after observing data

- **Likelihood**: \( p(y|X, w) \)
  - Measure of “fitness”

- **Prior density**: \( p(w) \)
  - Anything we know about parameters before we see any data

- **Marginal likelihood**: \( p(y|X) \)
  - It is a normalization constant – ensures \( \int p(w|X, y) dw = 1 \).
Bayesian Linear Regression - Finding posterior parameters

• Ignoring normalizing constants, the posterior is:

\[ p(w|X, y, \sigma^2) \propto \exp \left\{ -\frac{1}{2} (w - \mu)^\top \Sigma^{-1} (w - \mu) \right\} \]

\[ = \exp \left\{ -\frac{1}{2} (w^\top \Sigma^{-1} w - 2w^\top \Sigma^{-1} \mu + \mu^\top \Sigma^{-1} \mu) \right\} \]

\[ \propto \exp \left\{ -\frac{1}{2} (w^\top \Sigma^{-1} w - 2w^\top \Sigma^{-1} \mu) \right\} \]
• Ignoring non-$w$ terms, the prior multiplied by the likelihood is:

$$p(y|w, X, \sigma^2) \propto \exp\left\{-\frac{1}{2\sigma^2}(y - Xw)^\top(y - Xw)\right\} \exp\left\{-\frac{1}{2}w^\top S^{-1}w\right\}$$

$$\propto \exp\left\{-\frac{1}{2}\left(w^\top \left[\frac{1}{\sigma^2}X^\top X + S^{-1}\right]w - \frac{2}{\sigma^2}w^\top X^\top y\right)\right\}$$

• Posterior (from previous slide):

$$\propto \exp\left\{-\frac{1}{2}(w^\top \Sigma^{-1}w - 2w^\top \Sigma^{-1}\mu)\right\}$$
Bayesian Linear Regression - Finding posterior parameters

- Equate individual terms on each side.

- Covariance:

  \[
  \begin{align*}
  w^\top \Sigma^{-1}w &= w^\top \left[ \frac{1}{\sigma^2} X^\top X + S^{-1} \right] w \\
  \Sigma &= \left( \frac{1}{\sigma^2} X^\top X + S^{-1} \right)^{-1}
  \end{align*}
  \]

- Mean:

  \[
  \begin{align*}
  2w^\top \Sigma^{-1} \mu &= \frac{2}{\sigma^2} w^\top X^\top y \\
  \mu &= \frac{1}{\sigma^2} \Sigma X^\top y
  \end{align*}
  \]
Bayesian Linear Regression

- Posterior **must be** Gaussian

\[ p(w|X, y, \sigma^2) = \mathcal{N}(\mu, \Sigma) \]

- Covariance:

\[ \Sigma = \left( \frac{1}{\sigma^2} X^\top X + S^{-1} \right)^{-1} \]

- Mean:

\[ \mu = \frac{1}{\sigma^2} \Sigma X^\top y \]

- Predictions – same tedious exercise as before:

\[ p(y_*|X, y, X_*, \sigma^2) = \mathcal{N}(X_*^\top \mu, \sigma^2 + X_*^\top \Sigma X_*) \]
• Imagine transforming the inputs using a set of $D$ functions
\[ x \rightarrow \varphi(x) = (\varphi_1(x), \ldots, \varphi_D(x))^\top \]

• The functions $\varphi_1(X)$ are also known as basis functions

• Define:
\[ \Phi = \begin{bmatrix}
\varphi_1(x_1) & \cdots & \varphi_D(x_1) \\
\vdots & \ddots & \vdots \\
\varphi_1(x_N) & \cdots & \varphi_D(x_N)
\end{bmatrix} \]
Introducing basis functions

- Applying Bayesian Linear Regression on the transformed features gives

\[ p(w|X, y, \sigma^2) = \mathcal{N}(\mu, \Sigma) \]

- Covariance:

\[ \Sigma = \left( \frac{1}{\sigma^2} \Phi^\top \Phi + S^{-1} \right)^{-1} \]

- Mean:

\[ \mu = \frac{1}{\sigma^2} \Sigma \Phi^\top y \]

- Predictions:

\[ p(y_*|X, y, X_*, \sigma^2) = \mathcal{N}(\phi_*^\top \mu, \sigma^2 + \phi_*^\top \Sigma \phi_*) \]
Gaussian Processes
Gaussian Processes

- Linear models require specifying a set of basis functions
  - Polynomials, Trigonometric, ...??
Gaussian Processes

- Linear models require specifying a set of basis functions
  - Polynomials, Trigonometric, . . . ?
- Can we use Bayesian inference to let data tell this to us?
Gaussian Processes

- Linear models require specifying a set of basis functions
  - Polynomials, Trigonometric, \ldots？?
- Can we use Bayesian inference to let data tell this to us?
- Gaussian Processes work implicitly with an infinite set of basis functions and learn a probabilistic combination of these
• We are going to show that predictions can be expressed exclusively in terms of scalar products as follows

\[ k(x_i, x_j) = \psi(x_i)^\top \psi(x_j) \]

• This allows us to work with either \( k(\cdot, \cdot) \) or \( \psi(\cdot) \)

• Why is this useful??
Bayesian Linear Regression as a Kernel Machine

- Working with $\psi(\cdot)$ costs $O(D^2)$ storage, $O(D^3)$ time
- Working with $k(\cdot, \cdot)$ costs $O(N^2)$ storage, $O(N^3)$ time
• Working with $\psi(\cdot)$ costs $O(D^2)$ storage, $O(D^3)$ time
• Working with $k(\cdot, \cdot)$ costs $O(N^2)$ storage, $O(N^3)$ time
• Pick the one that makes computations faster . . . or
Bayesian Linear Regression as a Kernel Machine

- Working with $\psi(\cdot)$ costs $O(D^2)$ storage, $O(D^3)$ time
- Working with $k(\cdot, \cdot)$ costs $O(N^2)$ storage, $O(N^3)$ time
- Pick the one that makes computations faster . . . or
- What if we could pick $k(\cdot, \cdot)$ so that $\psi(\cdot)$ is infinite dimensional?
• It is possible to show that for

\[ k(x_i, x_j) = \exp \left( -\frac{\|x_i - x_j\|^2}{2} \right) \]

there exists a corresponding \( \psi(\cdot) \) that is infinite dimensional!!!

• There are other kernels satisfying this property
For simplicity consider one dimensional inputs $x_i$, $x_j$

Expand the Gaussian kernel $k(x_i, x_j)$ as

$$\exp \left( -\frac{(x_i - x_j)^2}{2} \right) = \exp \left( -\frac{x_i^2}{2} \right) \exp \left( -\frac{x_j^2}{2} \right) \exp (x_i x_j)$$

Focusing on the last term and applying the Taylor expansion of the $\exp(\cdot)$ function

$$\exp (x_i x_j) = 1 + (x_i x_j) + \frac{(x_i x_j)^2}{2!} + \frac{(x_i x_j)^3}{3!} + \frac{(x_i x_j)^4}{4!} + \ldots$$
• Define the infinite dimensional mapping

\[ \psi(x) = \exp \left( -\frac{x^2}{2} \right) \left( 1, x, \frac{x^2}{\sqrt{2!}}, \frac{x^3}{\sqrt{3!}}, \frac{x^4}{\sqrt{4!}}, \cdots \right)^\top \]

• It is easy to verify that

\[ k(x_i, x_j) = \exp \left( -\frac{(x_i - x_j)^2}{2} \right) = \psi(x_i)^\top \psi(x_j) \]
To show that Bayesian Linear Regression can be formulated through scalar products only, we need Woodbury identity:

\[(A + UCV)^{-1} = A^{-1} - A^{-1} U (C^{-1} + VA^{-1} U)^{-1} VA^{-1}\]

Intuitively:
• Woodbury identity:

$$(A + UCV)^{-1} = A^{-1} - A^{-1} U (C^{-1} + VA^{-1} U)^{-1} VA^{-1}$$

• We can rewrite:

$$\Sigma = \left( \frac{1}{\sigma^2} \Phi^T \Phi + S^{-1} \right)^{-1}$$

$$= S - S \Phi^T \left( \sigma^2 I + \Phi S \Phi^T \right)^{-1} \Phi S$$

• We set $A = S$, $U = V^\top = \Phi^\top$, and $C = \frac{1}{\sigma^2} I$
• Take $W^{(i)} \sim \mathcal{N}(0, \alpha_i I)$

• Central Limit Theorem implies that $f$ is Gaussian

• $f$ has zero-mean

• $\text{cov}(f) = E_{p(W^{(0)}, W^{(1)})} [\Phi(X W^{(0)}) W^{(1)} W^{(1)^\top} \Phi(X W^{(0)})^\top]$
Gaussian Processes as Infinitely-Wide Shallow Neural Nets

- Take $W^{(i)} \sim \mathcal{N}(0, \alpha_i I)$
- Central Limit Theorem implies that $f$ is Gaussian

- $f$ has zero-mean
- $\text{cov}(f) = \alpha_1 \mathbb{E}_p(W^{(0)}) [\Phi(XW^{(0)})\Phi(XW^{(0)})^\top]$  
- Some choices of $\Phi$ lead to analytic expression of known kernels (RBF, Matérn, arc-cosine, Brownian motion, ...)

Neal, LNS, 1996
Gaussian Processes for Regression

- Latent function:
  \[ f = w^\top \varphi(x) \]
  with \( \varphi(\cdot) \) possibly infinite dimensional!

- Bayes rule:
  \[
  p(f|X, y) = \frac{p(y|f)p(f|X)}{\int p(y|f)p(f|X)df} = \frac{p(y|f)p(f|X)}{p(y|X)}
  \]
Gaussian Processes for Regression

- Latent function:
  \[ f = \mathbf{w}^\top \varphi(\mathbf{x}) \]
  with \( \varphi(\cdot) \) possibly infinite dimensional!
- Bayes rule:
  \[
p(f|X, y) = \frac{p(y|f)p(f|X)}{\int p(y|f)p(f|X) df} = \frac{p(y|f)p(f|X)}{p(y|X)}
  \]
- Likelihood: \( p(y|f) = \mathcal{N}(y|\mathbf{0}, \sigma^2 I) \)
Gaussian Processes for Regression

- **Latent function:**
  \[ f = w^T \varphi(x) \]
  with \( \varphi(\cdot) \) possibly infinite dimensional!

- **Bayes rule:**
  \[
  p(f|X, y) = \frac{p(y|f)p(f|X)}{\int p(y|f)p(f|X)df} = \frac{p(y|f)p(f|X)}{p(y|X)}
  \]

- **Likelihood**: \( p(y|f) = \mathcal{N}(y|0, \sigma^2 I) \)

- **Prior over latent variables**: Implied by the prior over \( w \)
  \[
  p(f|X) = \mathcal{N}(f|0, K)
  \]
Gaussian Processes for Regression

- Latent function:
  \[ f = \mathbf{w}^\top \varphi(\mathbf{x}) \]
  with \( \varphi(\cdot) \) possibly infinite dimensional!
- Bayes rule:
  \[
p(f|X, y) = \frac{p(y|f)p(f|X)}{\int p(y|f)p(f|X)df} = \frac{p(y|f)p(f|X)}{p(y|X)}
  \]
- Likelihood: \( p(y|f) = \mathcal{N}(y|0, \sigma^2 I) \)
- Prior over latent variables: Implied by the prior over \( \mathbf{w} \)
  \[ p(f|X) = \mathcal{N}(f|0, K) \]
- Marginal likelihood: \( p(y|X) = \mathcal{N}(y|0, K + \sigma^2 I) \)
The kernel has parameters that have to be tuned

\[ k(x_i, x_j) = \alpha \exp(-\beta \| x_i - x_j \|^2) \]

\[ \ldots \text{and there is also the noise parameter } \sigma^2. \]

Define \( \theta = (\alpha, \beta, \sigma^2) \)

How should we tune them?
Optimization of Gaussian Process parameters

- Define $K_y = K + \sigma^2 I$
- Maximize the logarithm of the likelihood

$$p(y|X, \theta) = \mathcal{N}(0, K_y)$$

that is

$$-\frac{1}{2} \log|K_y| - \frac{1}{2} y^\top K_y^{-1} y + \text{const.}$$

- Derivatives can be useful for gradient-based optimization

$$\frac{\partial \log[p(y|X, \theta)]}{\partial \theta_i}$$
Optimization of Gaussian Process parameters

- Log-likelihood

\[-\frac{1}{2} \log |K_y| - \frac{1}{2} y^\top K_y^{-1} y + \text{const.}\]

- Derivatives can be useful for gradient-based optimization:

\[
\frac{\partial \log[p(y|X, \theta)]}{\partial \theta_i} = -\frac{1}{2} \text{Tr} \left( K_y^{-1} \frac{\partial K_y}{\partial \theta_i} \right) + \frac{1}{2} y^\top K_y^{-1} \frac{\partial K_y}{\partial \theta_i} K_y^{-1} y
\]
Challenges

- Non-Gaussian Likelihoods?
- Scalability?
• Marginal likelihood

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

can only be computed if \( p(y|f) \) is Gaussian

• What if \( p(y|f) \) is not Gaussian?
Tackling non-Gaussian case

- Approximation options:
  - Local variational bounds (classification only)
Tackling non-Gaussian case

- **Approximation options:**
  - Local variational bounds (classification only)
  - Laplace Approximation
    - Williams and Barber, *IEEE TPAMI*, 1998
Tackling non-Gaussian case

- Approximation options:
  - Local variational bounds (classification only)
  - Laplace Approximation
    - Williams and Barber, *IEEE TPAMI*, 1998
  - Expectation Propagation
  - Variational Bayes
    - Nickisch and Rasmussen, *JMLR*, 2008
    - Opper and Archambeau, *Neural Comp*, 2009
  - Markov chain Monte Carlo
    - Murray and Adams, *NIPS*, 2010
    - Filippone and Girolami, *IEEE TPAMI*, 2014
Tackling non-Gaussian case

- Approximation options:
  - Local variational bounds (classification only)
  - Laplace Approximation
    - Williams and Barber, *IEEE TPAMI*, 1998
  - Expectation Propagation
  - Variational Bayes
    - Nickisch and Rasmussen, *JMLR*, 2008
    - Opper and Archambeau, *Neural Comp*, 2009

- Markov chain Monte Carlo
  - Murray and Adams, *NIPS*, 2010
  - Filippone and Girolami, *IEEE TPAMI*, 2014
Tackling non-Gaussian case

- Approximation options:
  - Local variational bounds (classification only)
  - Laplace Approximation
    - Williams and Barber, *IEEE TPAMI*, 1998
  - Expectation Propagation
  - Variational Bayes
    - Nickisch and Rasmussen, *JMLR*, 2008
    - Opper and Archambeau, *Neural Comp*, 2009
  - Markov chain Monte Carlo
    - Murray and Adams, *NIPS*, 2010
    - Filippone and Girolami, *IEEE TPAMI*, 2014
Scalable Gaussian Processes
Marginal likelihood in GP models - Gaussian case and $n \gg$

- Marginal likelihood

$$p(y|X, \text{par}) = \int p(y|f)p(f|X, \text{par})df$$

can only be computed if $p(y|X, f)$ is Gaussian

- ... even then

$$\log[p(y|X, \text{par})] = -\frac{1}{2} \log |K_y| - \frac{1}{2} y^T K_y^{-1} y + \text{const.}$$

where $K_y = K(X, \text{par})$ is a $n \times n$ dense matrix!

- Complexity of exact method is $O(n^3)$ time and $O(n^2)$ space!
Tackling Gaussian case and $n \gg$

- Low-Rank Approximation options - $O(nm^2)$
- Call $P$ as a low rank approximation to $K_y$
- Woodbury identity exploits low rank structure of $P$

\[
K_y = \begin{bmatrix} \text{square} \end{bmatrix} + \begin{bmatrix} \text{diagonal} \end{bmatrix} \\

P = \begin{bmatrix} \text{upper triangular} \end{bmatrix} + \begin{bmatrix} \text{lower triangular} \end{bmatrix} \\

P^{-1} = \begin{bmatrix} \text{upper triangular} \end{bmatrix}^{-1} - \begin{bmatrix} \text{diagonal} \end{bmatrix}^{-1} \begin{bmatrix} \text{square} \end{bmatrix}^{-1} + \begin{bmatrix} \text{lower triangular} \end{bmatrix}^{-1}
\]
Tackling Gaussian case and $n \gg$

- Low-Rank Approximation options - $O(nm^2)$
  - Subset-of-data 'sparse' methods
    - Smola and Bartlett, *NIPS*, 2001
    - Seeger and Williams, *AISTATS*, 2003
- Pseudo-inputs introduced
  - Snelson and Ghahramani, *NIPS*, 2005
- A unifying view brings several ideas together
  - Quiñonero-Candela and Rasmussen, *JMLR*, 2005
- Variational approach for better placement of pseudo points
  - Titsias, *AISTATS*, 2009
- Random feature expansions
  - Rahimi and Recht, *NIPS*, 2008
  - Lazaro-Gredilla et al., *JMLR*, 2010
Tackling Gaussian case and $n \gg$ 

- Low-Rank Approximation options - $O(nm^2)$
  - Subset-of-data 'sparse' methods
    - Smola and Bartlett, *NIPS*, 2001
    - Seeger and Williams, *AISTATS*, 2003
  - Pseudo-inputs introduced
    - Snelson and Ghahramani, *NIPS*, 2005
Tackling Gaussian case and $n \gg$ 

- **Low-Rank Approximation options** - $O(nm^2)$
  - Subset-of-data 'sparse' methods
    - Smola and Bartlett, *NIPS*, 2001
    - Seeger and Williams, *AISTATS*, 2003
  - Pseudo-inputs introduced
    - Snelson and Ghahramani, *NIPS*, 2005
  - A unifying view brings several ideas together
    - Quiñonero-Candela and Rasmussen, *JMLR*, 2005

- **Random feature expansions**
  - Rahimi and Recht, *NIPS*, 2008
  - Lazaro-Gredilla et al., *JMLR*, 2010
Tackling Gaussian case and $n \gg$

- Low-Rank Approximation options - $O(nm^2)$
  - Subset-of-data 'sparse' methods
    - Smola and Bartlett, NIPS, 2001
    - Seeger and Williams, AISTATS, 2003
  - Pseudo-inputs introduced
    - Snelson and Ghahramani, NIPS, 2005
  - A unifying view brings several ideas together
    - Quiñonero-Candela and Rasmussen, JMLR, 2005
  - Variational approach for better placement of pseudo points
    - Titsias, AISTATS, 2009
Tackling Gaussian case and $n \gg$

- **Low-Rank Approximation options** - $O(nm^2)$
  - Subset-of-data 'sparse' methods
    - Smola and Bartlett, *NIPS*, 2001
    - Seeger and Williams, *AISTATS*, 2003
  - Pseudo-inputs introduced
    - Snelson and Ghahramani, *NIPS*, 2005
  - A unifying view brings several ideas together
    - Quiñonero-Candela and Rasmussen, *JMLR*, 2005
  - Variational approach for better placement of pseudo points
    - Titsias, *AISTATS*, 2009
  - Random feature expansions
    - Rahimi and Recht, *NIPS*, 2008
    - Lazaro-Gredilla et al., *JMLR*, 2010
Tackling Gaussian case and $n \gg$

- Approximation options:
  - Structured approximations based on Toeplitz/circulant matrices - $O\left( d n^{\frac{d+1}{d}} \right)$ time
    - Gilboa et al., *IEEE TPAMI*, 2015
  - Stochastic-gradient optimization/inference without model approximations - $O\left( n^2 \right)$ time and $O\left( n \right)$ space
    - Filippone and Engler, *ICML*, 2015
    - Cutajar, Osborne, Cunnningham, Filippone, *ICML*, 2016
Tackling Gaussian case and $n \gg$

- Approximation options:
  - Structured approximations based on Toeplitz/circulant matrices - $O(dn^{\frac{d+1}{d}})$ time
    - Gilboa et al., *IEEE TPAMI*, 2015
  - Stochastic-gradient optimization/inference **without** model approximations - $O(n^2)$ time and $O(n)$ space
    - Filippone and Engler, *ICML*, 2015
    - Cutajar, Osborne, Cunningham, Filippone, *ICML*, 2016
Challenges

- Non-Gaussian Likelihoods?
- Scalability?

Modern GP works tackle both
Desirable properties

- Mini-batch-based learning - $O(1)$ time for each iteration!
- Exploit GPU and distributed computing
- Automatic differentiation
- Application-specific representations (e.g., convolutional)
Stochastic Gradient Optimization

\[ E \left\{ \tilde{\nabla} \text{objective} \right\} = \nabla \text{objective} \]

Robbins and Monro, AoMS, 1951
Stochastic Gradient Optimization

\[ \text{par}' = \text{par} + \frac{\alpha_t}{2} \nabla \text{objective} \quad \alpha_t \to 0 \]

Robbins and Monro, *AoMS*, 1951
Modern GPs - Any likelihood and $n \gg$

- Approximation options:
  - Scalable Expectation Propagation
    - Bui et al., *ICML*, 2016
• Approximation options:
  ● Scalable Expectation Propagation
    ● Bui et al., *ICML*, 2016
  ● Inducing points methods
    ● Hensman et al., *AISTATS*, 2013
    ● Hensman, Matthews, Ghahramani, Filippone, *NIPS*, 2015
• Approximation options:
  • Scalable Expectation Propagation
    • Bui et al., *ICML*, 2016
  • Inducing points methods
    • Hensman et al., *AISTATS*, 2013
    • Hensman, Matthews, Ghahramani, Filippone, *NIPS*, 2015
  • Random feature expansions
    • Gal, Ghahramani, *ICML*, 2016
    • Cutajar, Bonilla, Michiardi, Filippone, *ICML*, 2017
EEG dataset
\((n = 14979, \, d = 14)\)
Teaser - Modern GPs - Any likelihood and $n \gg$

- Composition of processes - Deep Gaussian Processes

$$(f \circ g)(x)$$
Teaser - Modern GPs - Any likelihood and $n \gg$

- Composition of processes - Deep Gaussian Processes

 Damianou and Lawrence, AISTATS, 2013 – Cutajar, Bonilla, Michiardi, Filippone, ICML, 2017
Deep Gaussian processes on large datasets

**Airline dataset**

\(n = 5M+, \ d = 8\)
Other interesting topics

- Bayesian Optimization
  - Jones et al., *JoGO*, 1998
Other interesting topics

- Bayesian Optimization
  - Jones et al., JoGO, 1998

- Unsupervised Learning
  - Lawrence, NIPS, 2004
Other interesting topics

- Bayesian Optimization
  - Jones et al., *JoGO*, 1998
- Unsupervised Learning
  - Lawrence, *NIPS*, 2004
- Deep Gaussian Processes
  - Damianou and Lawrence, *AISTATS*, 2013
  - Cutajar, Bonilla, Michiardi, and Filippone, *ICML*, 2017
Other interesting topics

- Bayesian Optimization
  - Jones et al., *JoGO*, 1998
- Unsupervised Learning
  - Lawrence, *NIPS*, 2004
- Deep Gaussian Processes
  - Damianou and Lawrence, *AISTATS*, 2013
  - Cutajar, Bonilla, Michiardi, and Filippone, *ICML*, 2017
- Convolutional Gaussian Processes
  - Wilson et al., *AISTATS*, 2015
  - Wilson et al., *NIPS*, 2016
  - van der Wilk et al., *NIPS*, 2017
- Structured output
  - Galliani et al., *AISTATS*, 2017
- Probabilistic Numerics
  - Fitsimons, Cutajar, Osborne, Roberts, Filippone, *UAI*, 2017
Other interesting topics

- Bayesian Optimization
  - Jones et al., *JoGO*, 1998
- Unsupervised Learning
  - Lawrence, *NIPS*, 2004
- Deep Gaussian Processes
  - Damianou and Lawrence, *AISTATS*, 2013
  - Cutajar, Bonilla, Michiardi, and Filippone, *ICML*, 2017
- Convolutional Gaussian Processes
  - Wilson et al., *AISTATS*, 2015
  - Wilson et al., *NIPS*, 2016
  - van der Wilk et al., *NIPS*, 2017
- Structured output
  - Galliani et al., *AISTATS*, 2017
Other interesting topics

- Bayesian Optimization
  - Jones et al., *JoGO*, 1998
- Unsupervised Learning
  - Lawrence, *NIPS*, 2004
- Deep Gaussian Processes
  - Damianou and Lawrence, *AISTATS*, 2013
  - Cutajar, Bonilla, Michiardi, and Filippone, *ICML*, 2017
- Convolutional Gaussian Processes
  - Wilson et al., *AISTATS*, 2015
  - Wilson et al., *NIPS*, 2016
  - van der Wilk et al., *NIPS*, 2017
- Structured output
  - Galliani et al., *AISTATS*, 2017
- Probabilistic Numerics
  - Fitsimons, Cutajar, Osborne, Roberts, Filippone, *UAI*, 2017
Acknowledgments

Thank you!
Appendix
Bayesian Linear Regression as a Kernel Machine

- To show that Bayesian Linear Regression can be formulated through scalar products only, we need Woodbury identity:

\[(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}\]

- Intuitively:
Bayesian Linear Regression as a Kernel Machine

- Woodbury identity:

\[(A + UCV)^{-1} = A^{-1} - A^{-1} U(C^{-1} + VA^{-1}U)^{-1} VA^{-1}\]

- We can rewrite:

\[
\Sigma = \left( \frac{1}{\sigma^2} \Phi^\top \Phi + S^{-1} \right)^{-1}
\]

\[
= S - S\Phi^\top \left( \sigma^2 \mathbf{I} + \Phi S \Phi^\top \right)^{-1} \Phi S
\]

- We set \(A = S\), \(U = V^\top = \Phi^\top\), and \(C = \frac{1}{\sigma^2} \mathbf{I}\)
Bayesian Linear Regression as a Kernel Machine

- Mean and variance of the predictions:

\[
p(y_* | X, y, X_*, \sigma^2) = \mathcal{N}(\phi_*^T \mu, \sigma^2 + \phi_*^T \Sigma \phi_*)
\]

- Rewrite the variance:

\[
\sigma^2 + \phi_*^T \Sigma \phi_* = \\
\sigma^2 + \phi_*^T S \phi_* - \phi_*^T \Phi^T \left(\sigma^2 I + \Phi S \Phi^T \right)^{-1} \Phi S \phi_*
\]

... continued
Bayesian Linear Regression as a Kernel Machine

- Mean and variance of the predictions:

\[ p(y_\ast | X, y, X_\ast, \sigma^2) = \mathcal{N}(\phi_\ast^\top \mu, \sigma^2 + \phi_\ast^\top \Sigma \phi_\ast) \]

- Rewrite the variance:

\[ \sigma^2 + \phi_\ast^\top S \phi_\ast - \phi_\ast^\top S \phi^\top (\sigma^2 I + \Phi S \Phi^\top)^{-1} \Phi S \phi_\ast = \]

\[ \sigma^2 + k_{\ast\ast} - k_\ast^\top (\sigma^2 I + K)^{-1} k_\ast \]

- Where the mapping defining the kernel is

\[ \psi(X) = S^{1/2} \phi(X) \]

and

\[ k_{\ast\ast} = k(X_\ast, X_\ast) = \psi(X_\ast)^\top \psi(X_\ast) \]

\[ (k_\ast)_i = k(X_\ast, X_i) = \psi(X_\ast)^\top \psi(X_i) \]

\[ (K)_{ij} = k(X_i, X_j) = \psi(X_i)^\top \psi(X_j) \]
Bayesian Linear Regression as a Kernel Machine

- Mean and variance of the predictions:

\[ p(y_*|X, y, X_*, \sigma^2) = \mathcal{N}(\phi_\star^\top \mu, \sigma^2 + \phi_\star^\top \Sigma \phi_\star) \]

- Rewrite the mean:

\[
\begin{align*}
\phi_\star^\top \mu &= \frac{1}{\sigma^2} \phi_\star^\top \Sigma \Phi^\top y \\
&= \frac{1}{\sigma^2} \phi_\star^\top \left( S - S\Phi^\top \left( \sigma^2 I + \Phi S\Phi^\top \right)^{-1} \Phi S \right) \Phi^\top y \\
&= \frac{1}{\sigma^2} \phi_\star^\top S\Phi^\top \left( I - \left( \sigma^2 I + \Phi S\Phi^\top \right)^{-1} \Phi S\Phi^\top \right) y \\
&= \frac{1}{\sigma^2} \phi_\star^\top S\Phi^\top \left( I - \left( I + \frac{\Phi S\Phi^\top}{\sigma^2} \right)^{-1} \frac{\Phi S\Phi^\top}{\sigma^2} \right) y
\end{align*}
\]

... continued
Bayesian Linear Regression as a Kernel Machine

• Define $H = \frac{\Phi S \Phi^T}{\sigma^2}$

• The term in the parenthesis

$$
\left( I - \left( I + \frac{\Phi S \Phi^T}{\sigma^2} \right)^{-1} \frac{\Phi S \Phi^T}{\sigma^2} \right)
$$

becomes

$$
\left( I - (I + H)^{-1} H \right) = I - (H^{-1} + I)^{-1}
$$

• Using Woodbury ($A, U, V = I$ and $C = H^{-1}$)

$$
I - (H^{-1} + I)^{-1} = (I + H)^{-1}
$$
Bayesian Linear Regression as a Kernel Machine

- Substituting into the expression of the predictive mean

\[
\phi^\top_* \mu = \frac{1}{\sigma^2} \phi^\top_* S \phi^\top \left( I - \left( I + \frac{\Phi S \Phi^\top}{\sigma^2} \right)^{-1} \frac{\Phi S \Phi^\top}{\sigma^2} \right) y
\]

\[
= \frac{1}{\sigma^2} \phi^\top_* S \phi^\top \left( I + \frac{\Phi S \Phi^\top}{\sigma^2} \right)^{-1} y
\]

\[
= \phi^\top_* S \phi^\top \left( \sigma^2 I + \Phi S \Phi^\top \right)^{-1} y
\]

\[
= k^\top_* \left( \sigma^2 I + K \right)^{-1} y
\]

- All definitions as in the case of the variance

\[
\psi(X) = S^{1/2} \phi(X)
\]

\[
(k_*),_i = k(X_*, X_i) = \psi(X_*)^\top \psi(X_i)
\]

\[
(K)_{ij} = k(X_i, X_j) = \psi(X_i)^\top \psi(X_j)
\]