Gaussian processes and Bayesian NNs in function space

Simón Rodriguez Santana

JAE school of mathematics 2021
Institute of Mathematical Sciences ICMAT-CSIC

First of all... Why all this?

Modern *machine learning* $\rightarrow$ mostly centered on **point-wise predictions**

First of all... Why all this?

Modern *machine learning* $\rightarrow$ mostly centered on **point-wise predictions**

Estimate the uncertainty of the predictions $\rightarrow$ **Bayesian approach** in the model formulation

First of all... Why all this?

Modern *machine learning* $\rightarrow$ mostly centered on **point-wise predictions**

Estimate the uncertainty of the predictions $\rightarrow$ **Bayesian approach** in the model formulation

Posterior Dist. $p(\mathbf{W}|\text{Data}) = p(\mathbf{W})p(\text{Data}|\mathbf{W})/p(\text{Data})$

Predictive Dist. $p(y|\text{Data}, x) = \int p(y|\mathbf{W}, x)p(\mathbf{W}|\text{Data})d\mathbf{W}$

First of all... Why all this?

Modern *machine learning* $\rightarrow$ mostly centered on **point-wise predictions**

Estimate the uncertainty of the predictions $\rightarrow$ **Bayesian approach** in the model formulation

Posterior Dist. $$p(\mathbf{W}|\text{Data}) = p(\mathbf{W})p(\text{Data}|\mathbf{W})/p(\text{Data})$$

Predictive Dist. $$p(y|\text{Data}, x) = \int p(y|\mathbf{W}, x)p(\mathbf{W}|\text{Data})d\mathbf{W}$$

Computing $p(\text{Data})$ is intractable! $\Rightarrow$ different **approximate solutions**, such as **BNNs** (*VI, EP, AVB, etc.*) or **GPs**

Non-paremetric approaches *s.a.* **GPs** could help ease our job (real-world problems are complicated!)

$\rightarrow$ *Intrinsic advantages and issues!*

Ideal case

We would like a model capable capable of:

Ideal case

We would like a model capable capable of:

1. Producing **flexible predictive distributions** so it's useful in many different problems (reproduce *exotic* behavior *s.a.* bimodality, heterocedasticity, etc.)

Ideal case

We would like a model capable capable of:

1. Producing **flexible predictive distributions** so it's useful in many different problems (reproduce *exotic* behavior *s.a.* bimodality, heterocedasticity, etc.)
2. Bayesian approach $\Rightarrow$ **Sensibly update the prior** *w.r.t.* the training data

Ideal case

We would like a model capable capable of:

1. Producing **flexible predictive distributions** so it's useful in many different problems (reproduce *exotic* behavior *s.a.* bimodality, heterocedasticity, etc.)
2. Bayesian approach $\Rightarrow$ **Sensibly update the prior** *w.r.t.* the training data
3. Being **scalable to large datasets**

Ideal case

We would like a model capable capable of:

1. Producing **flexible predictive distributions** so it's useful in many different problems (reproduce *exotic* behavior *s.a.* bimodality, heterocedasticity, etc.)
2. Bayesian approach ⇒ **Sensibly update the prior** *w.r.t.* the training data
3. Being **scalable to large datasets**

Stablished methods ⇒ lack some properties, while exceed at others

Could we combine some of them to improve overall?

Brief mention of kernel methods

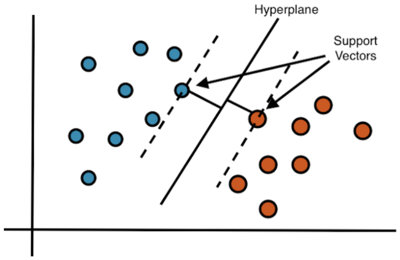
- Widespread models based on learning **kernel functions**
- **Instance based methods** $\Rightarrow$ Learn parameters for each training data point (*must remember these*)
- **Predictions** $\Rightarrow$ Similarity function $k(\cdot, \cdot)$ between train and test points (**kernel**)
- Kernel can be decomposed by a *feature space* mapping $\phi(\cdot)$

Brief mention of kernel methods

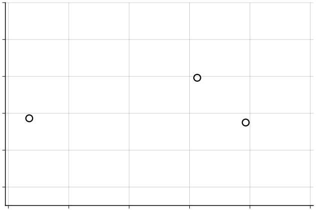
- Widespread models based on learning **kernel functions**
- **Instance based methods** $\Rightarrow$ Learn parameters for each training data point (*must remember these*)
- **Predictions** $\Rightarrow$ Similarity function $k(\cdot,\cdot)$ between train and test points (**kernel**)
- Kernel can be decomposed by a *feature space* mapping $\phi(\cdot)$

$$k(x,x') = \phi(x)^T\phi(x')$$

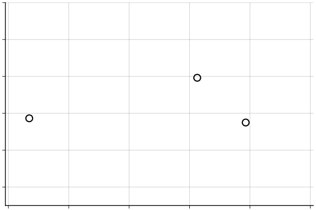
- Many different kernels to choose from
- Flexible approach $\Rightarrow$ many different usages (SVMs, GPs, PCA...)

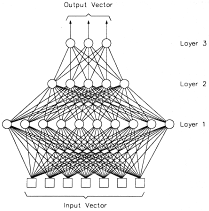


Approximate inference and GPs



Approximate inference and GPs

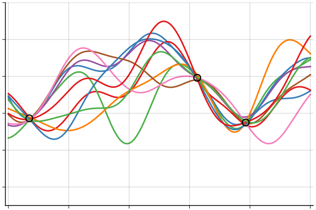


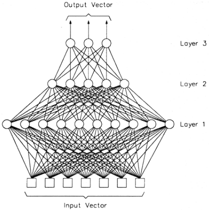


$$h_j(\mathbf{x}) = \tanh\left(\sum_{i=1}^{I} x_i w_{ji}\right)$$

$$f(\mathbf{x}) = \sum_{j=1}^{H} v_j h_j(\mathbf{x})$$

Approximate inference and GPs

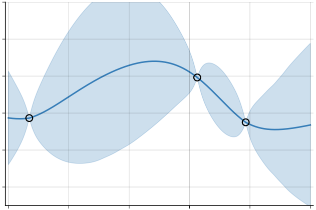


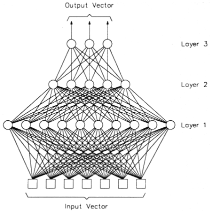


$$h_j(\mathbf{x}) = \tanh\left(\sum_{i=1}^{I} x_i w_{ji}\right)$$

$$f(\mathbf{x}) = \sum_{j=1}^{H} v_j h_j(\mathbf{x})$$

Approximate inference and GPs

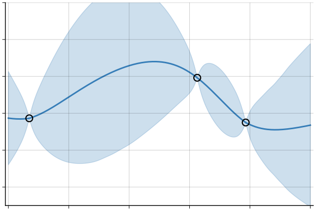


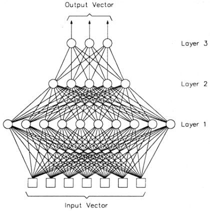


$$h_j(\mathbf{x}) = \tanh\left(\sum_{i=1}^{I} x_i w_{ji}\right)$$

$$f(\mathbf{x}) = \sum_{j=1}^{H} v_j h_j(\mathbf{x})$$

Approximate inference and GPs





$$h_j(\mathbf{x}) = \tanh\left(\sum_{i=1}^{I} x_i w_{ji}\right)$$

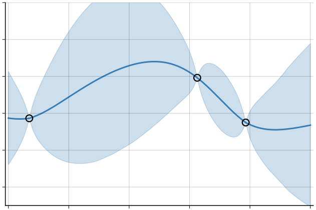
$$f(\mathbf{x}) = \sum_{j=1}^{H} v_j h_j(\mathbf{x})$$

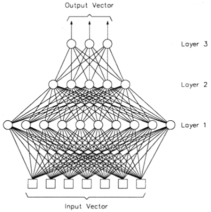
Posterior Dist. $p(\mathbf{W}|\text{Data}) = p(\mathbf{W})p(\text{Data}|\mathbf{W})/p(\text{Data})$

Predictive Dist. $p(y|\text{Data}, x) = \int p(y|\mathbf{W}, x)p(\mathbf{W}|\text{Data})d\mathbf{W}$

Challenges: Non-parametric models would simplify our job (problems can be complex!) and computing $p(\text{Data})$ is intractable!

Approximate inference and GPs





$$h_j(\mathbf{x}) = \tanh\left(\sum_{i=1}^{I} x_i w_{ji}\right)$$

$$f(\mathbf{x}) = \sum_{j=1}^{H} v_j h_j(\mathbf{x})$$

Posterior Dist. $p(\mathbf{W}|\text{Data}) = p(\mathbf{W})p(\text{Data}|\mathbf{W})/p(\text{Data})$

Predictive Dist. $p(y|\text{Data}, x) = \int p(y|\mathbf{W}, x)p(\mathbf{W}|\text{Data})d\mathbf{W}$

Challenges: Non-parametric models would simplify our job (problems can be complex!) and computing $p(\text{Data})$ is intractable!

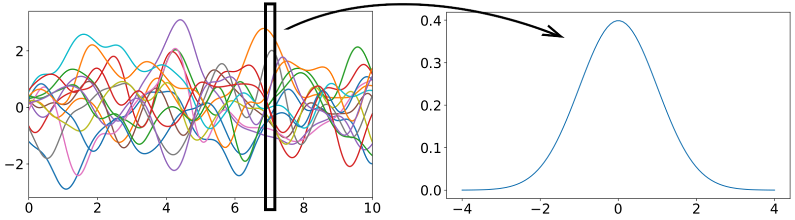
Hint: One (*vanilla*) solution is simply setting $p(\mathbf{W}) \sim \mathcal{N}(\mathbf{W}|0, \sigma^2\mathbf{I})$

Gaussian Processes

GPs: Distribution over functions $f(\cdot)$ so that for any finite $\{\mathbf{x}_i\}_{i=1}^N$, $(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N))^\mathsf{T}$ follows an N-dimensional Gaussian distribution.

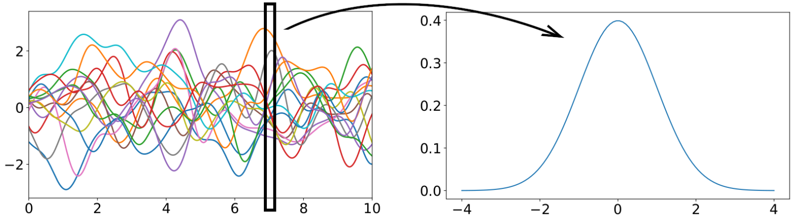
Gaussian Processes

GPs: Distribution over functions $f(\cdot)$ so that for any finite $\{\mathbf{x}_i\}_{i=1}^N$, $(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N))^\top$ follows an N-dimensional Gaussian distribution.



Gaussian Processes

GPs: Distribution over functions $f(\cdot)$ so that for any finite $\{\mathbf{x}_i\}_{i=1}^N$, $(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N))^\mathsf{T}$ follows an N-dimensional Gaussian distribution.

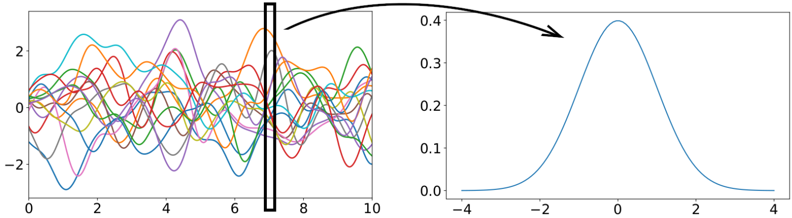


Regression with GPs

$$\hat{y}_i = y_i + \epsilon_i, \qquad \text{with} \qquad p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}), \quad \epsilon_i \sim \mathcal{N}(0, \beta^{-1})$$

Gaussian Processes

GPs: Distribution over functions $f(\cdot)$ so that for any finite $\{\mathbf{x}_i\}_{i=1}^{N}$, $(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N))^\mathsf{T}$ follows an N-dimensional Gaussian distribution.



Regression with GPs

$$\hat{y}_i = y_i + \epsilon_i, \qquad \text{with} \qquad p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}), \quad \epsilon_i \sim \mathcal{N}(0, \beta^{-1})$$

Due to Gaussian form, there are **closed-form solutions** for many useful questions about finite data!

Gaussian Processes

- The **joint distribution** for $\mathbf{y}^\star$ at test points $\{\mathbf{x}_m^\star\}_{m=1}^M$ and $\mathbf{y}$:

$$p(\mathbf{y}^\star, \mathbf{y}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\kappa}_\theta & \mathbf{k}_\theta^\mathsf{T} \\ \mathbf{k}_\theta & \mathbf{K}_\theta \end{bmatrix}\right)$$

Gaussian Processes

- The **joint distribution** for $\mathbf{y}^\star$ at test points $\{\mathbf{x}^\star_m\}_{m=1}^M$ and $\mathbf{y}$:

$$p(\mathbf{y}^\star, \mathbf{y}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\kappa}_\theta & \mathbf{k}_\theta^\mathsf{T} \\ \mathbf{k}_\theta & \mathbf{K}_\theta \end{bmatrix} \right)$$

- These **matrices** are computed from the covariance $C(\cdot, \cdot; \theta)$:

$$[\mathbf{K}_\theta]_{n,n'} = C(\mathbf{x}_n, \mathbf{x}_{n'}; \theta)$$

$$[\mathbf{k}_\theta]_{n,m} = C(\mathbf{x}_n, \mathbf{x}^\star_m; \theta), \qquad [\boldsymbol{\kappa}_\theta]_{m,m'} = C(\mathbf{x}^\star_m, \mathbf{x}^\star_{m'}; \theta),$$

Gaussian Processes

- The **joint distribution** for $\mathbf{y}^\star$ at test points $\{\mathbf{x}_m^\star\}_{m=1}^M$ and $\mathbf{y}$:

$$p(\mathbf{y}^\star, \mathbf{y}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\kappa}_\theta & \mathbf{k}_\theta^\top \\ \mathbf{k}_\theta & \mathbf{K}_\theta \end{bmatrix} \right)$$

- These **matrices** are computed from the covariance $C(\cdot, \cdot; \theta)$:

$$[\mathbf{K}_\theta]_{n,n'} = C(\mathbf{x}_n, \mathbf{x}_{n'}; \theta)$$

$$[\mathbf{k}_\theta]_{n,m} = C(\mathbf{x}_n, \mathbf{x}_m^\star; \theta)\,, \qquad [\boldsymbol{\kappa}_\theta]_{m,m'} = C(\mathbf{x}_m^\star, \mathbf{x}_{m'}^\star; \theta)\,,$$

- The **predictive distribution** for $\mathbf{y}^\star$ given $\mathbf{y}$, $p(\mathbf{y}^\star|\mathbf{y})$, is:

$$\mathbf{y}^\star \sim \mathcal{N}(\mathbf{m}, \boldsymbol{\Sigma})$$

$$\mathbf{m} = \mathbf{k}_\theta^\top \mathbf{K}_\theta^{-1} \mathbf{y}\,, \qquad \boldsymbol{\Sigma} = \boldsymbol{\kappa}_\theta - \mathbf{k}_\theta^\top \mathbf{K}_\theta^{-1} \mathbf{k}_\theta\,,$$

Gaussian Processes

- The **joint distribution** for $\mathbf{y}^\star$ at test points $\{\mathbf{x}_m^\star\}_{m=1}^M$ and $\mathbf{y}$:

$$p(\mathbf{y}^\star, \mathbf{y}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\kappa}_\theta & \mathbf{k}_\theta^\mathsf{T} \\ \mathbf{k}_\theta & \mathbf{K}_\theta \end{bmatrix} \right)$$

- These **matrices** are computed from the covariance $C(\cdot, \cdot; \theta)$:

$$[\mathbf{K}_\theta]_{n,n'} = C(\mathbf{x}_n, \mathbf{x}_{n'}; \theta)$$

$$[\mathbf{k}_\theta]_{n,m} = C(\mathbf{x}_n, \mathbf{x}_m^\star; \theta)\,, \qquad [\boldsymbol{\kappa}_\theta]_{m,m'} = C(\mathbf{x}_m^\star, \mathbf{x}_{m'}^\star; \theta)\,,$$

- The **predictive distribution** for $\mathbf{y}^\star$ given $\mathbf{y}$, $p(\mathbf{y}^\star|\mathbf{y})$, is:

$$\mathbf{y}^\star \sim \mathcal{N}(\mathbf{m}, \boldsymbol{\Sigma})$$

$$\mathbf{m} = \mathbf{k}_\theta^\mathsf{T} \mathbf{K}_\theta^{-1} \mathbf{y}\,, \qquad \boldsymbol{\Sigma} = \boldsymbol{\kappa}_\theta - \mathbf{k}_\theta^\mathsf{T} \mathbf{K}_\theta^{-1} \mathbf{k}_\theta\,,$$

- The log of the **marginal likelihood**, $p(\mathbf{y}|\theta)$, is:

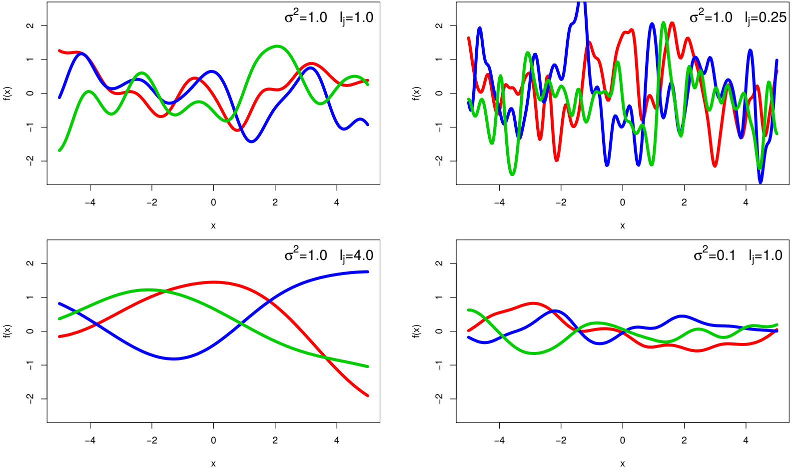
$$\log p(\mathbf{y}) = -\frac{N}{2} \log 2\pi - \frac{1}{2} \log |\mathbf{K}_\theta| - \frac{1}{2} \mathbf{y}^\mathsf{T} \mathbf{K}_\theta^{-1} \mathbf{y}$$

An Example of a Covariance Function

Squared Exponential: $$C(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left\{\frac{1}{2}\sum_{j=1}^{d}\left(\frac{x_j - x_j'}{l_j}\right)^2\right\}$$

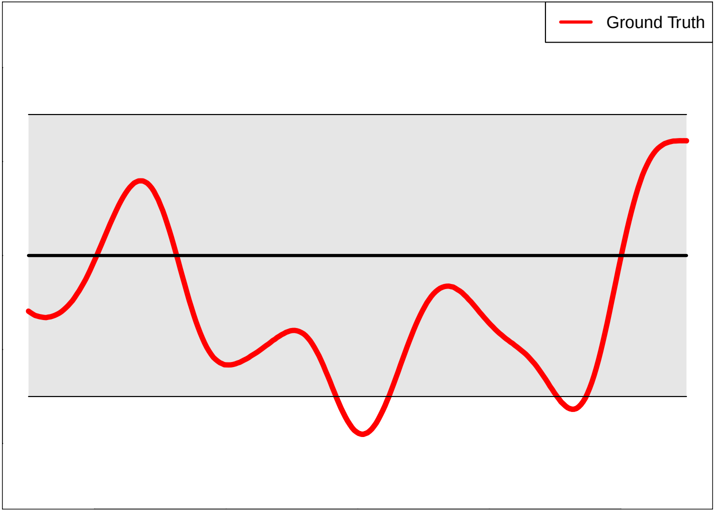
An Example of a Covariance Function

Squared Exponential: $$C(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left\{\frac{1}{2}\sum_{j=1}^{d}\left(\frac{x_j - x_j'}{l_j}\right)^2\right\}$$



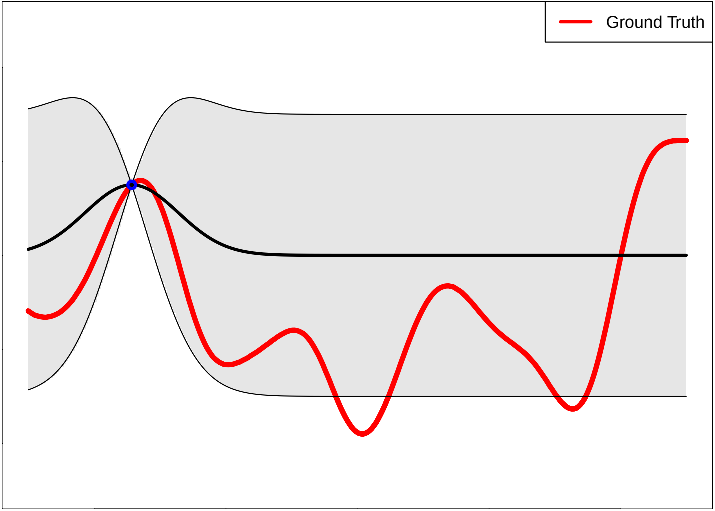
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



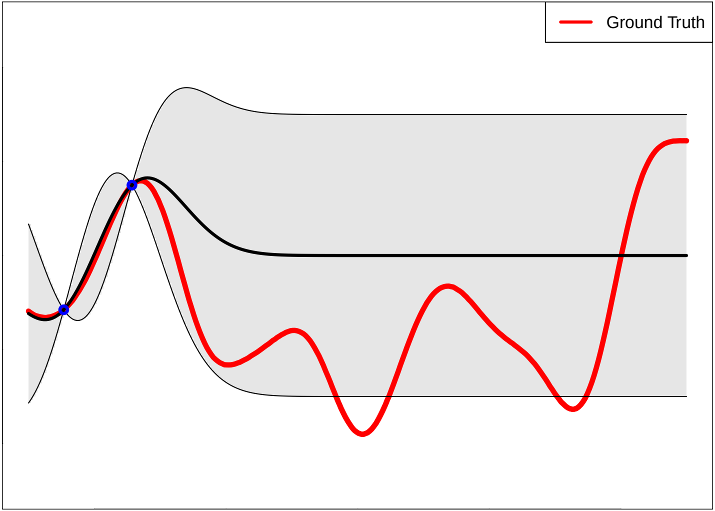
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



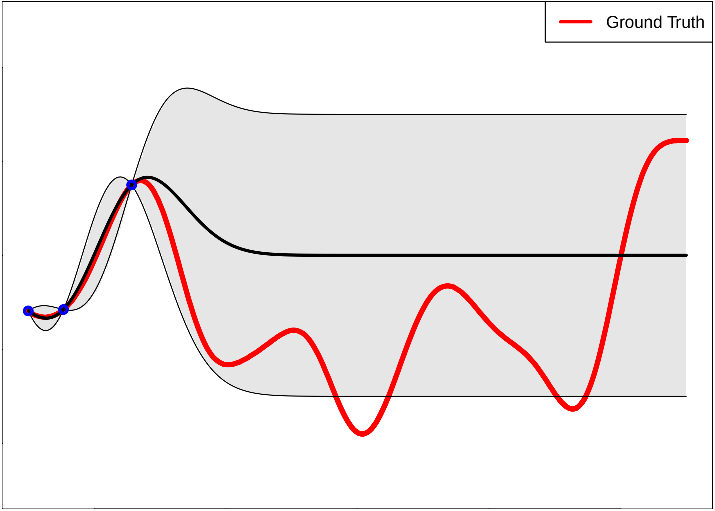
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



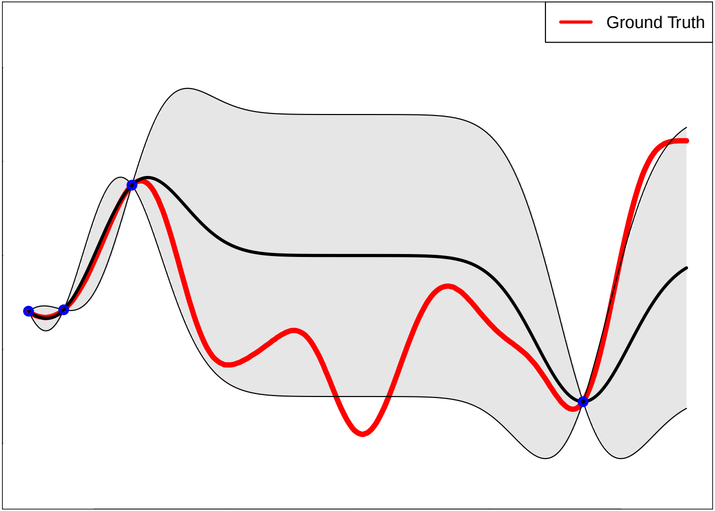
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



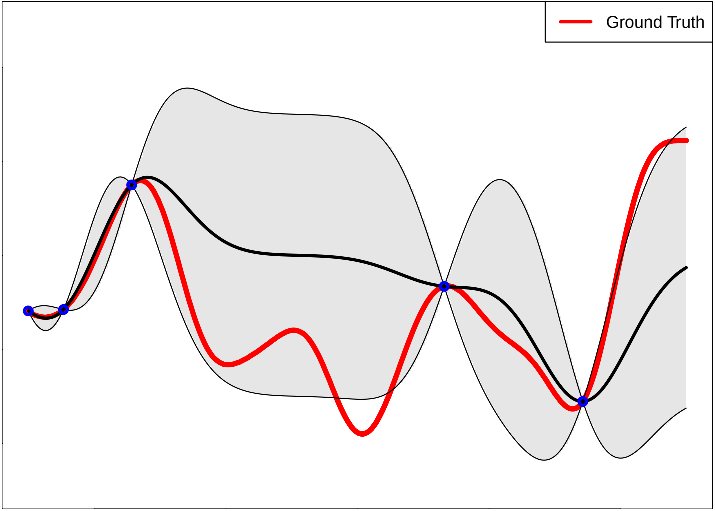
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



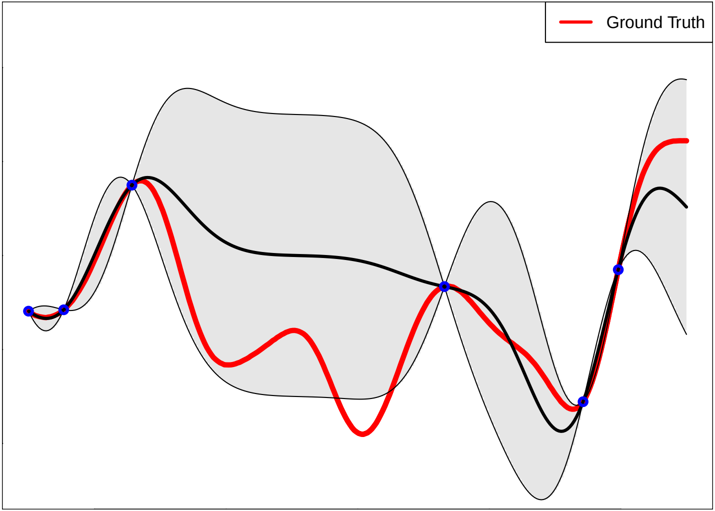
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



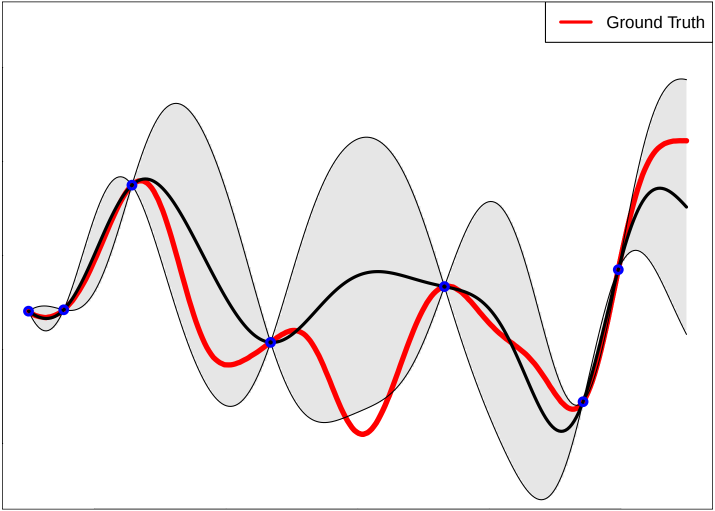
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



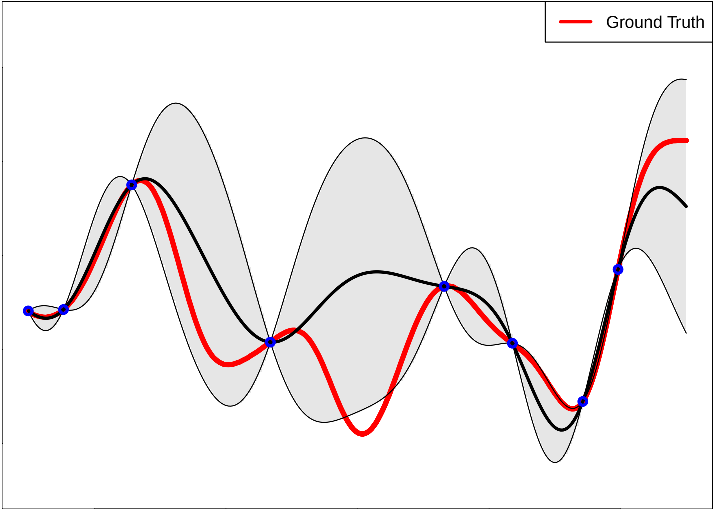
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



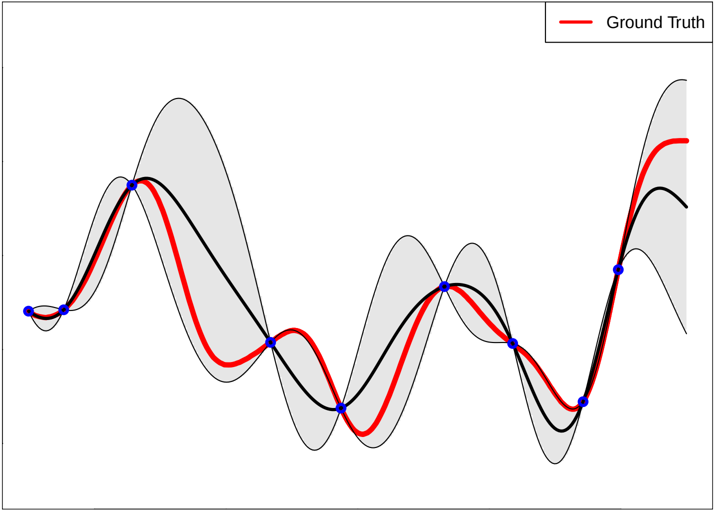
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



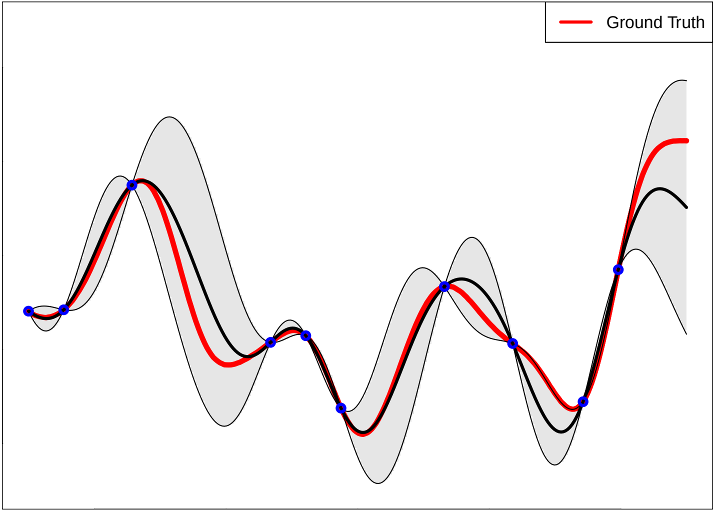
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



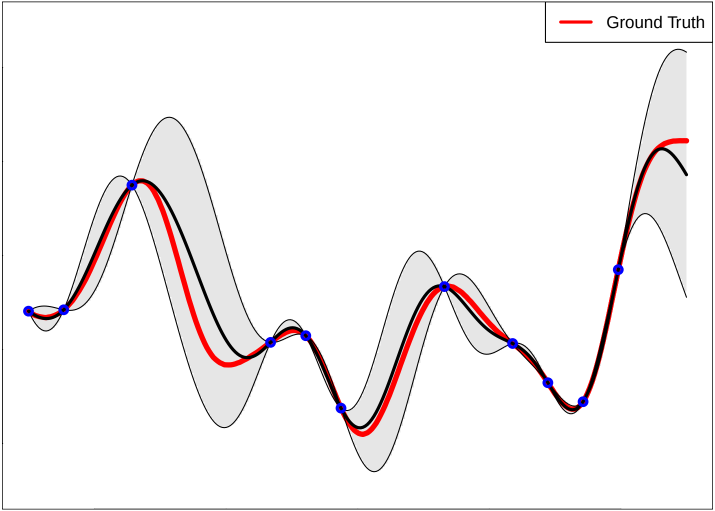
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



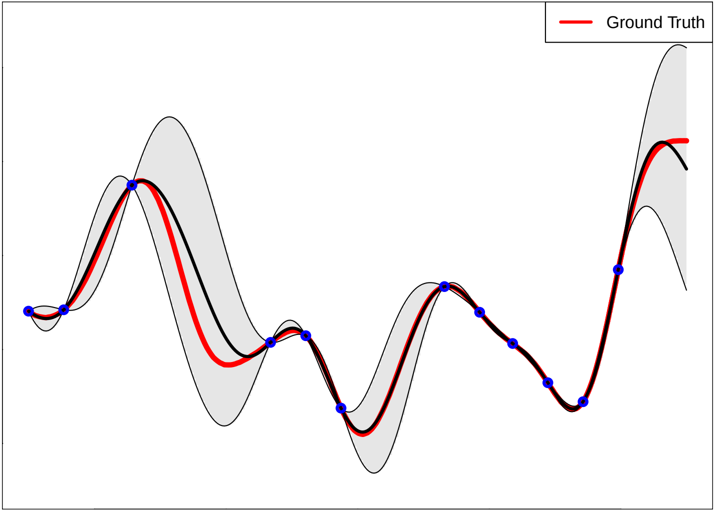
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



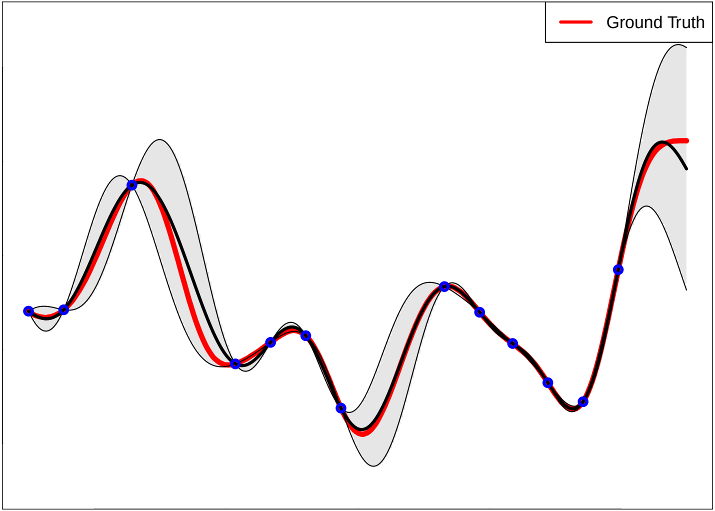
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



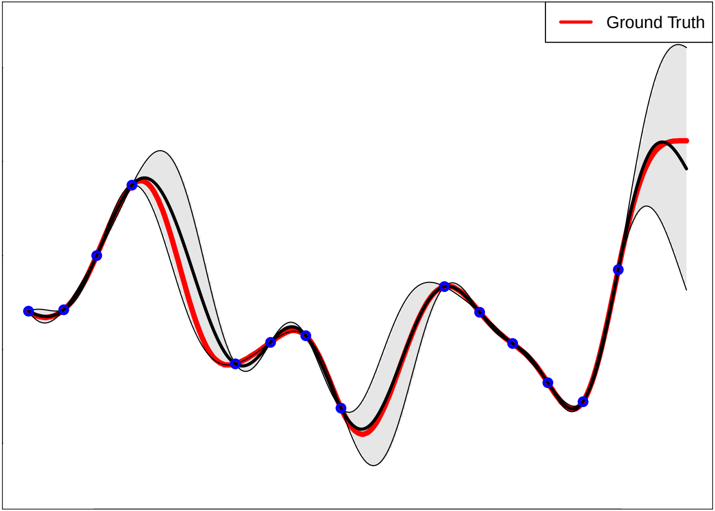
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



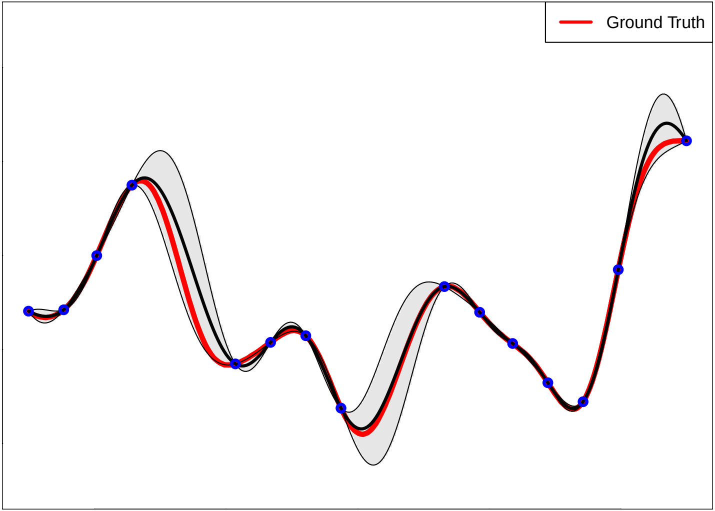
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



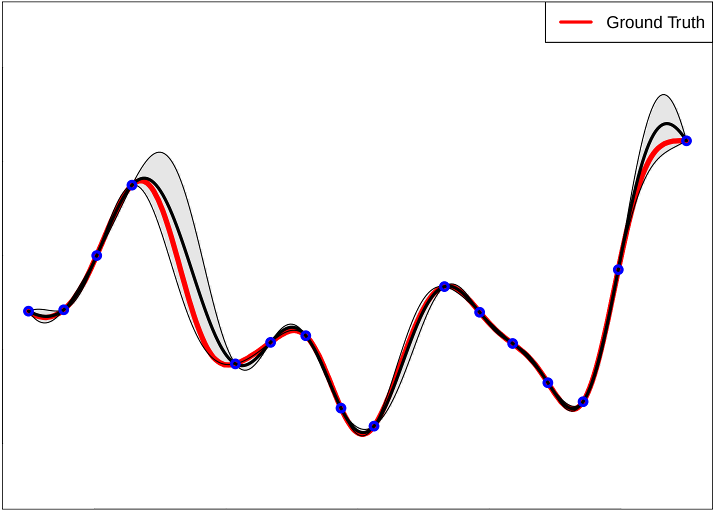
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



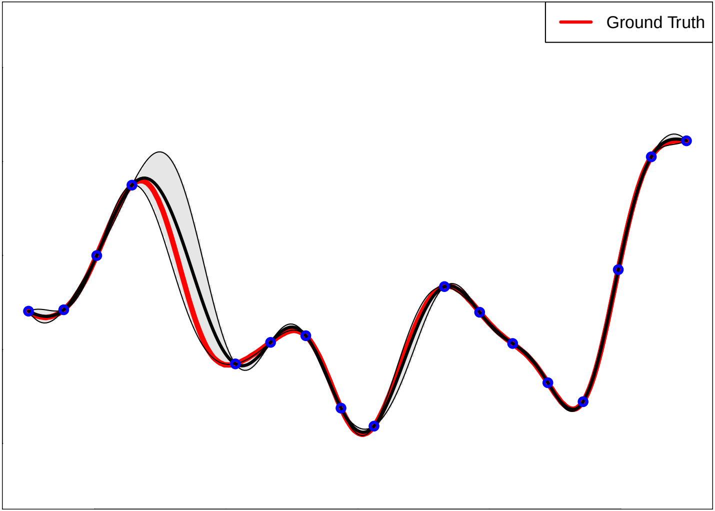
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



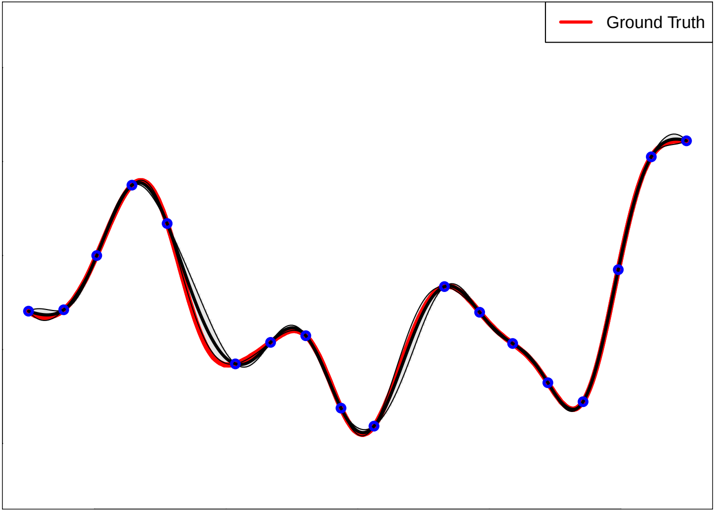
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



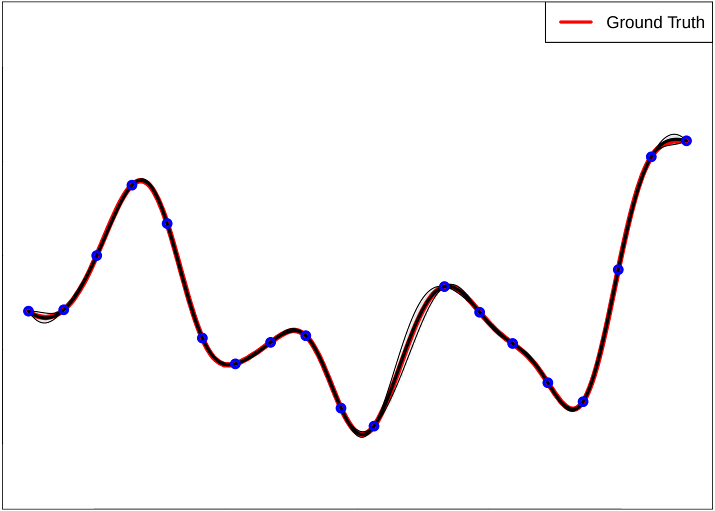
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



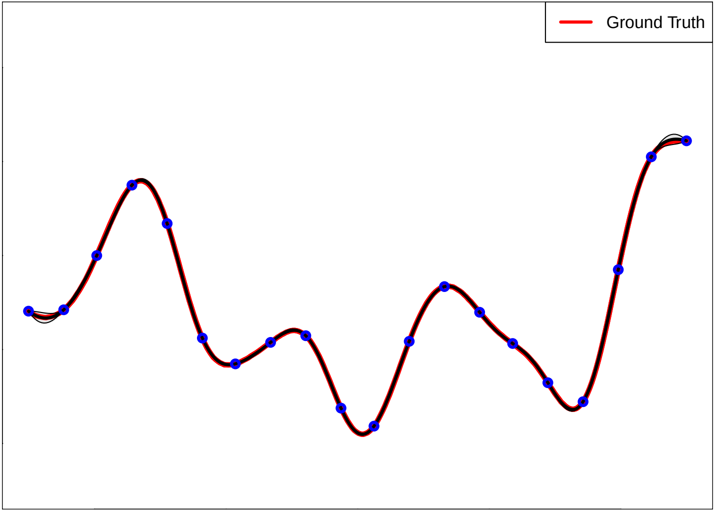
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.



Some issues with GPs

GPs are an easy approach for regression in simple issues

- Non-parametric.
- Bayesian inference is tractable.
- Hyper-parameter tuning by maximizing the marginal likelihood.

Some issues with GPs

GPs are an easy approach for regression in simple issues

- Non-parametric.
- Bayesian inference is tractable.
- Hyper-parameter tuning by maximizing the marginal likelihood.

They also have important **problems**:

- **Scalability**: Matrix inversion ($\mathbf{K}_\theta^{-1}$) is super costly ($\mathcal{O}(N^3)$)!
- **Gaussianity**: Normal behavior may be too simple for real-world problems!

Some issues with GPs

GPs are an easy approach for regression in simple issues

- Non-parametric.
- Bayesian inference is tractable.
- Hyper-parameter tuning by maximizing the marginal likelihood.

They also have important **problems**:

- **Scalability**: Matrix inversion ($\mathbf{K}_\theta^{-1}$) is super costly ($\mathcal{O}(N^3)$)!
- **Gaussianity**: Normal behavior may be too simple for real-world problems!

NNs are interesting as well

- Automatic feature representation learning.
- Scale to very large datasets.
- Bayesian inference is intractable.

Some issues with GPs

GPs are an easy approach for regression in simple issues

- Non-parametric.
- Bayesian inference is tractable.
- Hyper-parameter tuning by maximizing the marginal likelihood.

They also have important **problems**:

- **Scalability**: Matrix inversion ($\mathbf{K}_\theta^{-1}$) is super costly ($\mathcal{O}(N^3)$)!
- **Gaussianity**: Normal behavior may be too simple for real-world problems!

NNs are interesting as well

- Automatic feature representation learning.
- Scale to very large datasets.
- Bayesian inference is intractable.

Can we get the benefits of the two approaches?

Bayesian Neural Networks

Carry out approximate Bayesian inference in neural networks with a finite number of neurons in the space of weights!

Bayesian Neural Networks

Carry out approximate Bayesian inference in neural networks with a finite number of neurons in the space of weights!

Posterior Dist. $p(\mathbf{W}|\text{Data}) = p(\mathbf{W})p(\text{Data}|\mathbf{W})/p(\text{Data})$

Predictive Dist. $p(y|\text{Data}, x) = \int p(y|\mathbf{W}, x)p(\mathbf{W}|\text{Data})d\mathbf{W}$

Bayesian Neural Networks

Carry out approximate Bayesian inference in neural networks with a finite number of neurons in the space of weights!

Posterior Dist. $p(\mathbf{W}|\text{Data}) = p(\mathbf{W})p(\text{Data}|\mathbf{W})/p(\text{Data})$

Predictive Dist. $p(y|\text{Data}, x) = \int p(y|\mathbf{W}, x)p(\mathbf{W}|\text{Data})d\mathbf{W}$

How do we approximate these quantities?

Variational Inference - a quick reminder

Used to **find the parameters** of a distribution q, so that it **looks similar** to some target distribution p, known up to the normalization constant.

Variational Inference - a quick reminder

Used to **find the parameters** of a distribution q, so that it **looks similar** to some target distribution p, known up to the normalization constant.

It is based on the following **decomposition**:

$$\log p(\mathcal{D}) = \mathcal{L}(q) + \mathrm{KL}(q|p)$$

where

Variational Inference - a quick reminder

Used to **find the parameters** of a distribution q, so that it **looks similar** to some target distribution p, known up to the normalization constant.

It is based on the following **decomposition**:

$$\log p(\mathcal{D}) = \mathcal{L}(q) + \mathrm{KL}(q|p)$$

where

$$\mathcal{L}(q) = \int q(W) \log \left\{ \frac{p(\mathbf{W}, \mathcal{D})}{q(\mathbf{W})} \right\} d\mathbf{W}\,, \quad \mathrm{KL}(q||p) = -\int q(\mathbf{W}) \log \left\{ \frac{p(\mathbf{W}|\mathcal{D})}{q(\mathbf{W})} \right\} d\mathbf{W} \geq 0$$

Variational Inference - a quick reminder

Used to **find the parameters** of a distribution q, so that it **looks similar** to some target distribution p, known up to the normalization constant.

It is based on the following **decomposition**:

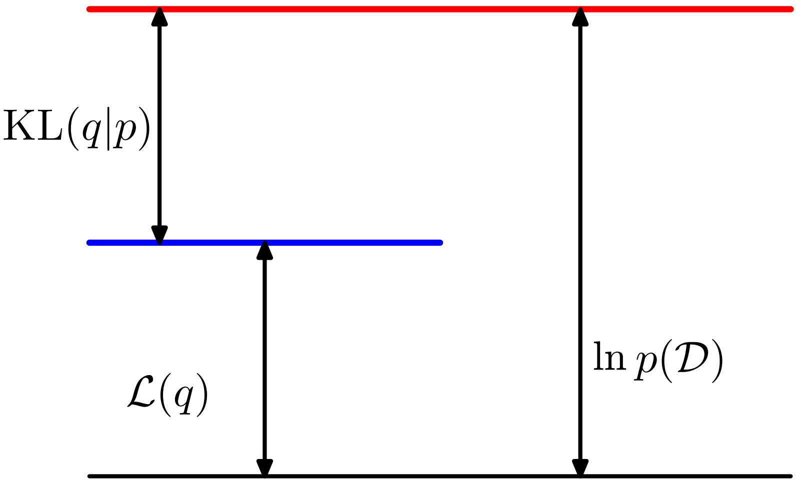
$$\log p(\mathcal{D}) = \mathcal{L}(q) + \mathrm{KL}(q|p)$$

where

$$\mathcal{L}(q) = \int q(\boldsymbol{W}) \log \left\{ \frac{p(\mathbf{W}, \mathcal{D})}{q(\mathbf{W})} \right\} d\mathbf{W}\,, \quad \mathrm{KL}(q||p) = -\int q(\mathbf{W}) \log \left\{ \frac{p(\mathbf{W}|\mathcal{D})}{q(\mathbf{W})} \right\} d\mathbf{W} \geq 0$$

$p(\mathbf{W}, \mathcal{D})$, the product of the prior and the likelihood factors, **simplifies with the logarithm** and $\mathcal{L}(q)$ is **feasible to evaluate**.

Decomposition of the Marginal Likelihood



Maximization of the Lower Bound

The joint distribution can be expressed as $p(\mathcal{D}, \mathbf{W}) = p(\mathcal{D}|\mathbf{W})p(\mathbf{W})$.

Maximization of the Lower Bound

The joint distribution can be expressed as $p(\mathcal{D}, \mathbf{W}) = p(\mathcal{D}|\mathbf{W})p(\mathbf{W})$.

If the likelihood **factorizes across data instances** $(y_i, \mathbf{x}_i)$:

$$\mathcal{L} = \sum_{i=1}^{N} \mathbb{E}_q[\log p(\mathbf{y}_i|\mathbf{W}, \mathbf{x}_i)] - \mathrm{KL}(q|\mathrm{prior})$$

Maximization of the Lower Bound

The joint distribution can be expressed as $p(\mathcal{D}, \mathbf{W}) = p(\mathcal{D}|\mathbf{W})p(\mathbf{W})$.

If the likelihood **factorizes across data instances** $(y_i, \mathbf{x}_i)$:

$$\mathcal{L} = \sum_{i=1}^{N} \mathbb{E}_q[\log p(\mathbf{y}_i|\mathbf{W}, \mathbf{x}_i)] - \mathrm{KL}(q|\mathrm{prior})$$

- Monte Carlo and mini-batches!
- Closed form solution if the prior and q are Gaussian!

Maximization of the Lower Bound

The joint distribution can be expressed as $p(\mathcal{D}, \mathbf{W}) = p(\mathcal{D}|\mathbf{W})p(\mathbf{W})$.

If the likelihood **factorizes across data instances** $(y_i, \mathbf{x}_i)$:

$$\mathcal{L} = \sum_{i=1}^{N} \mathbb{E}_q[\log p(\mathbf{y}_i|\mathbf{W}, \mathbf{x}_i)] - \mathrm{KL}(q|\mathrm{prior})$$

- Monte Carlo and mini-batches!
- Closed form solution if the prior and q are Gaussian!

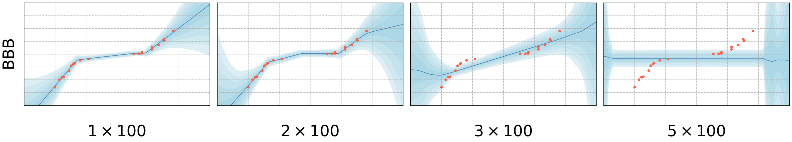
Stochastic optimization techniques enable VI on deep neural networks and massive datasets!

Approximate Inference in Weight Space

The posterior distribution is very complicated and q is often parametric and assumes independence!

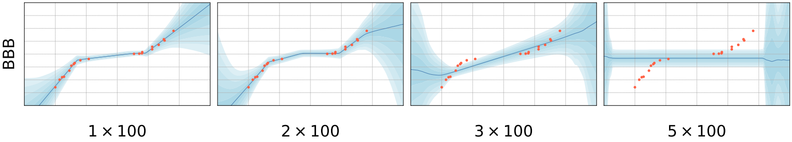
Approximate Inference in Weight Space

The posterior distribution is very complicated and q is often parametric and assumes independence!



Approximate Inference in Weight Space

The posterior distribution is very complicated and q is often parametric and assumes independence!



Undesirable behavior as more units or layers are added!

(Sun et al., 2019)

Why use the function space?

Benefits:

1. Avoids symmetric modes in the posterior of parameter space!
2. May potentially give better predictions and uncertainty estimates.
3. May consider more flexible priors than GPs.
4. Avoids pathologies related to the size of the inference problem!

Why use the function space?

Benefits:

1. Avoids symmetric modes in the posterior of parameter space!
2. May potentially give better predictions and uncertainty estimates.
3. May consider more flexible priors than GPs.
4. Avoids pathologies related to the size of the inference problem!

Approximate inference is challenging since it involves working with random functions rather than with finite sets of variables!

Implicit Processes

Collection of random variables $f(\cdot)$, such that any finite collection $\{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)\}$ has joint distribution defined by the generative process:

$$\mathbf{z} \sim p(\mathbf{z})\,, \qquad\qquad f(\mathbf{x}_n) = g_\theta(\mathbf{x}_n, \mathbf{z})\,.$$

Implicit Processes

Collection of random variables $f(\cdot)$, such that any finite collection $\{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)\}$ has joint distribution defined by the generative process:

$$\mathbf{z} \sim p(\mathbf{z})\,, \qquad f(\mathbf{x}_n) = g_\theta(\mathbf{x}_n, \mathbf{z})\,.$$

Bayesian neural networks: $\theta \Rightarrow$ means and variances of $\mathbf{W}$

$$\mathbf{W} \sim \mathcal{N}(\mathbf{W}|\mathbf{0}, \mathbf{I})\,, \qquad f(\mathbf{x}) = g_\theta(\mathbf{W}, \mathbf{x})\,,$$

Neural sampler: $\theta \Rightarrow$ weights of non-linear function $g_\theta(\cdot, \cdot)$.

Implicit Processes

Collection of random variables $f(\cdot)$, such that any finite collection $\{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)\}$ has joint distribution defined by the generative process:

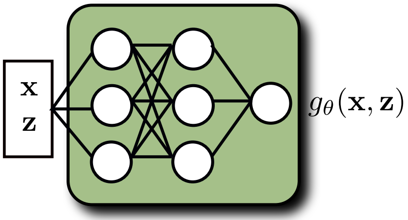
$$\mathbf{z} \sim p(\mathbf{z}), \qquad f(\mathbf{x}_n) = g_\theta(\mathbf{x}_n, \mathbf{z}).$$

Bayesian neural networks: $\theta \Rightarrow$ means and variances of $\mathbf{W}$

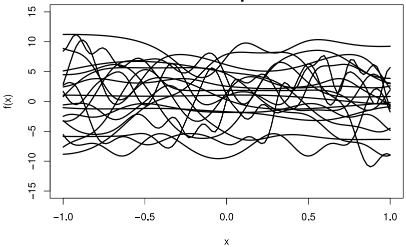
$$\mathbf{W} \sim \mathcal{N}(\mathbf{W}|\mathbf{0}, \mathbf{I}), \qquad f(\mathbf{x}) = g_\theta(\mathbf{W}, \mathbf{x}),$$

Neural sampler: $\theta \Rightarrow$ weights of non-linear function $g_\theta(\cdot, \cdot)$.

Neural sampler



Prior samples



Learning under Implicit Process Priors

Ideally we would like to satisfy two goals:

1. Find a flexible approximation for the posterior distribution.
2. Train the model's prior parameters θ.

Learning under Implicit Process Priors

Ideally we would like to satisfy two goals:

1. Find a flexible approximation for the posterior distribution.
2. Train the model's prior parameters θ.

Two approaches in the literature:

1. Variational Implicit Process (Ma et al., 2019).
   - Learns θ, but with Gaussian predictions
2. Functional Bayesian Neural Network (Sun et al., 2019).
   - Flexible implicit predictive distributions, but cannot learn θ.

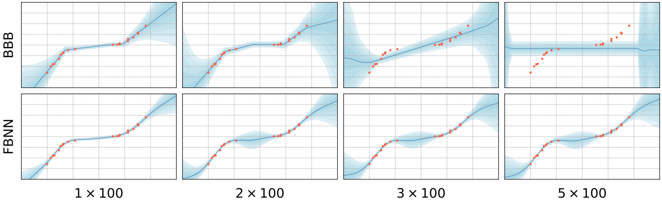
Learning under Implicit Process Priors

Ideally we would like to satisfy two goals:

1. Find a flexible approximation for the posterior distribution.
2. Train the model's prior parameters θ.

Two approaches in the literature:

1. Variational Implicit Process (Ma et al., 2019).
   - Learns θ, but with Gaussian predictions
2. Functional Bayesian Neural Network (Sun et al., 2019).
   - Flexible implicit predictive distributions, but cannot learn θ.



Inference with IPs and inducing points

Implicit process $f(\mathbf{x}) = h_\phi(\mathbf{x}, \epsilon)$ as approximate implicit posterior distribution of the process specified in the prior (as in *FBNNs*)

Approximate Inference via functional variational inference (*f-ELBO*):

$$\mathcal{L}(q) = \sum_{i=1}^{N} \mathbb{E}_q[\log p(y_i | f(\mathbf{x}_i))] - \mathrm{KL}(q | \mathrm{prior}) \,.$$

Inference with IPs and inducing points

Implicit process $f(\mathbf{x}) = h_\phi(\mathbf{x}, \epsilon)$ as approximate implicit posterior distribution of the process specified in the prior (as in *FBNNs*)

Approximate Inference via functional variational inference (*f-ELBO*):

$$\mathcal{L}(q) = \sum_{i=1}^{N} \mathbb{E}_q[\log p(y_i | f(\mathbf{x}_i))] - \mathrm{KL}(q|\mathrm{prior}) \,.$$

Challenges:

1. Avoid increasing the number of latent variables with N (as GPs)
   - $M \ll N$ **inducing points** ($\overline{\mathbf{X}}$, $\mathbf{u}$)
2. Compute the conditional posterior (intractable)
   - **MonteCarlo GP approximation** for the posterior approximation $p(\mathbf{f}|\mathbf{u})$ (as in *VIPs*)

Training the system

Our posterior approximation becomes

$$q(\mathbf{f}, \mathbf{u}) = p_\theta(\mathbf{f}|\mathbf{u}) q_\phi(\mathbf{u})$$

The variational inference objective is:

$$\begin{aligned}\mathcal{L}(q) &= \mathbb{E}_q \left[\log \frac{p(\mathbf{y}|\mathbf{f}) \cancel{p_\theta(\mathbf{f}|\mathbf{u})} p_\theta(\mathbf{u})}{\cancel{p_\theta(\mathbf{f}|\mathbf{u})} q_\phi(\mathbf{u})} \right] \\ &= \sum_{i=1}^{N} \mathbb{E}_{q_{\phi,\theta}} [\log p(y_i|f_i)] - \mathrm{KL}(q_\phi(\mathbf{u}) | p_\theta(\mathbf{u}))\end{aligned}$$

Training the system

Our posterior approximation becomes

$$q(\mathbf{f}, \mathbf{u}) = p_\theta(\mathbf{f}|\mathbf{u}) q_\phi(\mathbf{u})$$

The variational inference objective is:

$$\mathcal{L}(q) = \mathbb{E}_q \left[\log \frac{p(\mathbf{y}|\mathbf{f}) \cancel{p_\theta(\mathbf{f}|\mathbf{u})} p_\theta(\mathbf{u})}{\cancel{p_\theta(\mathbf{f}|\mathbf{u})} q_\phi(\mathbf{u})} \right]$$
$$= \sum_{i=1}^{N} \mathbb{E}_{q_{\phi,\theta}} [\log p(y_i|f_i)] - \mathrm{KL}(q_\phi(\mathbf{u})|p_\theta(\mathbf{u}))$$

KL-divergence is **intractable** (**implicit** q and p) $\Rightarrow$ **classifier** to estimate the log-ratio inside the KL-divergence:

$$\mathrm{KL}(q_\phi(\mathbf{u})|p_\theta(\mathbf{u})) = -\mathbb{E}_q \left[\log \frac{p_\theta(\mathbf{u})}{q_\phi(\mathbf{u})} \right] = -\mathbb{E}_q \left[T_{\Omega^\star}(\mathbf{u}) \right]$$

$T_{\Omega^\star}(\mathbf{u}) \Rightarrow$ Optimized DNN discriminating samples of $q_\phi(\mathbf{u})$ and $p_\theta(\mathbf{u})$

Conditional Distribution and Predictions

It is critical to compute $p_\theta(\mathbf{f}|\mathbf{u})$ in the model.

Conditional Distribution and Predictions

It is critical to compute $p_\theta(\mathbf{f}|\mathbf{u})$ in the model.

Approximated using a GP (as in VIP)

$$\mathbb{E}[f(\mathbf{x})] = m^\star_{MLE}(\mathbf{x}) + \mathbf{K}_{\mathbf{f},\mathbf{u}}(\mathbf{K}_{\mathbf{u},\mathbf{u}} + \mathbf{I}\sigma^2)^{-1}(\mathbf{u} - m^\star_{\mathrm{MLE}}(\mathbf{X}))\,,$$
$$\mathrm{Var}(f(\mathbf{x})) = \mathbf{K}_{\mathbf{f},\mathbf{f}} - \mathbf{K}_{\mathbf{f},\mathbf{f}}(\mathbf{K}_{\mathbf{u},\mathbf{u}} + \mathbf{I}\sigma^2)^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}$$

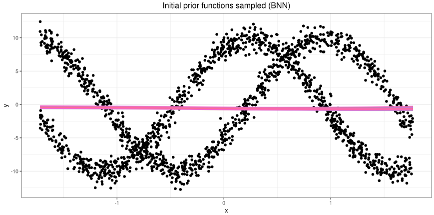
Covariances $\Rightarrow$ Monte Carlo methods by sampling from the prior

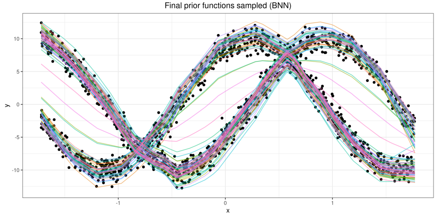
Predictions can also be approximated by Monte Carlo:

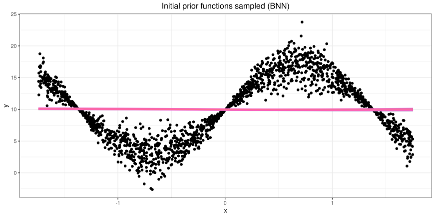
$$p(f(\mathbf{x}_*)|\mathbf{y},\mathbf{X}) \approx \frac{1}{S}\sum_{s=1}^{S} p_\theta(f(\mathbf{x}_*)|\mathbf{u}_s)\,, \qquad \mathbf{u} \sim q_\phi(\mathbf{u})\,.$$

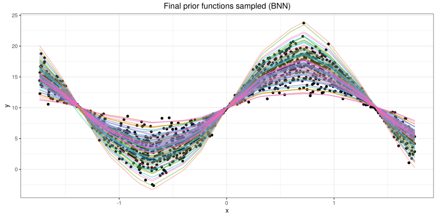
Flexibility of the prior functions

Synthetic data with different features to test the functions the prior is able to learn



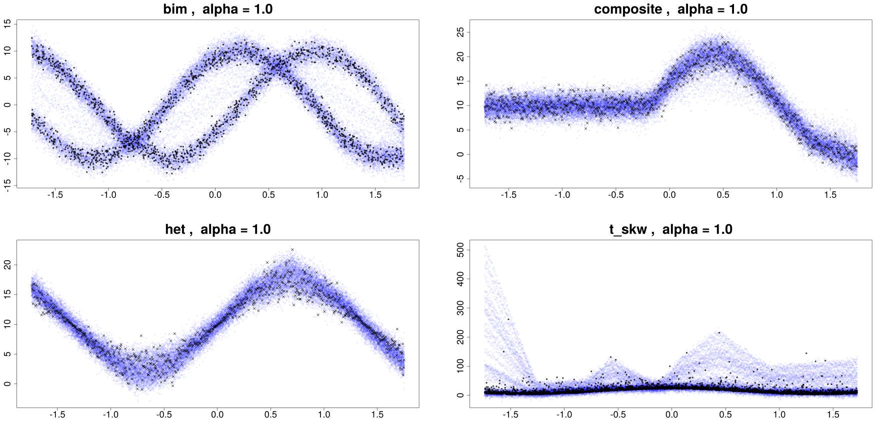






Predictive distribution and results

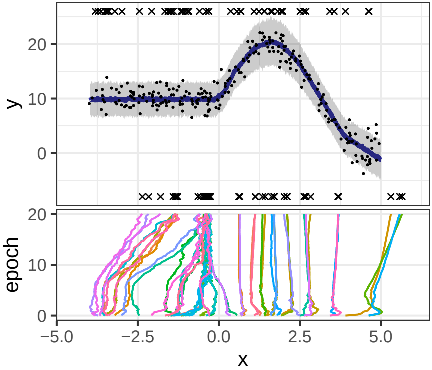
Flexible final predictions in different synthetic datasets



Evolution of the inducing points

Inducing points tend to gather in the regions where data changes most
The data here follows a constant function first, and suddenly change into a sine function

- The matching point between both behaviors tend to have more concentration of IPs ($M = 50$)



Conclusions

1. Gaussian Processes and Bayesian neural networks provide partial solutions for estimating uncertainty in the predictions.
   - GPs: simple and work fine for small data, but have flexibility and scalability problems
   - Sparse GPs: scalable, but predictions remain only Gaussian
   - BNNs: intractable inference and issues in the optimization procedure
2. Approximate inference in function space may be advantageous over weight space
3. Implicit processes are a difficult but very useful tool to deal with all these issues
   - Availability to **learn the hyperparameters** θ (*IP* prior) ✓
   - Flexibility in the posterior approximation (*IP* model - NS) with mixture of Gaussians predictions $\Rightarrow$ General predictive dist. ✓
   - **Scalability** in memory ($\mathcal{O}(M^3)$) and convergence time ✓

Thank you for your attention!

References

- Ma, C., Li, Y., Hernández-Lobato, J. M. Variational implicit processes. International Conference on Machine Learning, 2019.
- Titsias, M. (2009, April). Variational learning of inducing variables in sparse Gaussian processes. In Artificial Intelligence and Statistics (pp. 567-574).
- Snelson, E. and Ghahramani, Z. Sparse Gaussian processes using pseudo-inputs. NIPS 18, pp. 1257-1264, 2006.
- S. Sun, G. Zhang, J. Shi, R. Grosse. Functional Variational Bayesian Neural Networks. International Conference on Learning Representations, 2019.
- Mescheder, L., Nowozin, S., Geiger, A. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. International Conference on Machine Learning, 2017.
- Williams, C. K. I. and Barber, D. Bayesian classification with Gaussian processes. IEEE PAMI, 20,1342-1351, 1998.