Additive Gaussian Processes

David Duvenaud, Hannes Nickisch, Carl Rasmussen

Cambridge University
Computational and Biological Learning Lab

January 13, 2012
Outline

Gaussian Process Regression
  Definition
  Properties

Additive Gaussian Processes
  Central Modeling Assumption
  Interpretability
  Related Work
  Results
Given $X, y$, predict some new function value $y^*$ at location $x^*$. 
Regression Methods

Given $X, y$, predict some new function value $y^*$ at location $x^*$. Many methods with nice properties.
Regression Methods

Given $X, y$, predict some new function value $y^*$ at location $x^*$. Many methods with nice properties.

**Linear Regression** - Fast
Regression Methods

Given $X, y$, predict some new function value $y^*$ at location $x^*$. Many methods with nice properties.

**Linear Regression** - Fast

**Deep Belief Networks** - Semi-supervised
Regression Methods

Given $\mathbf{X}, \mathbf{y}$, predict some new function value $\mathbf{y}^*$ at location $\mathbf{x}^*$. Many methods with nice properties.

- **Linear Regression** - Fast
- **Deep Belief Networks** - Semi-supervised
- **Spline Models** - Nonparametric
Regression Methods

Given \( \mathbf{X}, \mathbf{y} \), predict some new function value \( \mathbf{y}^* \) at location \( \mathbf{x}^* \). Many methods with nice properties.

Linear Regression  -  Fast
Deep Belief Networks  -  Semi-supervised
Spline Models  -  Nonparametric
Gaussian Process Regression
Regression Methods

Given $X, y$, predict some new function value $y^*$ at location $x^*$. Many methods with nice properties.

- **Linear Regression** - Fast
- **Deep Belief Networks** - Semi-supervised
- **Spline Models** - Nonparametric
- **Gaussian Process Regression**
  - Non-parametric
Regression Methods

Given $X, y$, predict some new function value $y^*$ at location $x^*$. Many methods with nice properties.

- **Linear Regression** - Fast
- **Deep Belief Networks** - Semi-supervised
- **Spline Models** - Nonparametric
- **Gaussian Process Regression**
  - Non-parametric
  - Data-Efficient
Regression Methods

Given $\mathbf{X}, \mathbf{y}$, predict some new function value $\mathbf{y}^*$ at location $\mathbf{x}^*$. Many methods with nice properties.

**Linear Regression** - Fast  
**Deep Belief Networks** - Semi-supervised  
**Spline Models** - Nonparametric  
**Gaussian Process Regression**
  - Non-parametric  
  - Data-Efficient  
  - Tractable Joint Posterior
Definition

Assume our data \((X, y)\) is generated by \(y = f(x) + \epsilon_\sigma\)

\(f\) is a **latent function** which we need to do inference about.
Definition

Assume our data \((X, y)\) is generated by \(y = f(x) + \epsilon_\sigma\)

\(f\) is a **latent function** which we need to do inference about. A GP prior distribution over \(f\) means that, for any finite set of indices \(X\),

\[
p(f_x|\theta) = \mathcal{N}(\mu_\theta(X), K_\theta(X, X))
\]
Definition

Assume our data \((X, y)\) is generated by \(y = f(x) + \epsilon_\sigma\)

\(f\) is a **latent function** which we need to do inference about.

A GP prior distribution over \(f\) means that, for any finite set of indices \(X\),

\[
p(f_x | \theta) = \mathcal{N}(\mu_\theta(X), K_\theta(X, X))
\]

where

\[
K_{ij} = k_\theta(x, x')
\]

is the **covariance function** or **kernel**, which specifies the covariance between two function values \(f(x_1), f(x_2)\) given their locations \(x_1, x_2\).
Assume our data \((\mathbf{X}, \mathbf{y})\) is generated by \(\mathbf{y} = \mathbf{f}(\mathbf{x}) + \epsilon_\sigma\). 

\(\mathbf{f}\) is a \textbf{latent function} which we need to do inference about. 

A GP prior distribution over \(\mathbf{f}\) means that, for any finite set of indices \(\mathbf{X}\),

\[ p(\mathbf{f}_X | \theta) = \mathcal{N}(\mu_\theta(\mathbf{X}), K_\theta(\mathbf{X}, \mathbf{X})) \]

where

\[ K_{ij} = k_\theta(x, x') \]

is the \textit{covariance function} or \textit{kernel}, which specifies the covariance between two function values \(f(x_1), f(x_2)\) given their locations \(x_1, x_2\).

e.g.

\[ k_\theta(x, x') = \exp(-\frac{1}{2\theta}|x-x'|^2) \]
Sampling from a GP

```matlab
function simple_gp_sample
    % Choose a set of x locations.
    N = 100;
    x = linspace(-2, 2, N);

    % Specify the covariance between function values, depending on their location.
    for j = 1:N
        for k = 1:N
            sigma(j,k) = covariance( x(j), x(k) );
        end
    end

    % Specify that the prior mean of f is zero.
    mu = zeros(N, 1);

    % Sample from a multivariate Gaussian.
    f = mvnrnd( mu, sigma );

    plot(x, f);
end

% Squared-exp covariance function.
function k = covariance(x, y)
    k = exp( -0.5*( x - y )^2 );
end
```
Sampling from a GP

```matlab
function simple_gp_sample

    % Choose a set of x locations.
    N = 100;
    x = linspace(-2, 2, N);

    % Specify the covariance between function values, depending on their location.
    for j = 1:N
        for k = 1:N
            sigma(j,k) = covariance( x(j), x(k) );
        end
    end

    % Specify that the prior mean of f is zero.
    mu = zeros(N, 1);

    % Sample from a multivariate Gaussian.
    f = mvnrnd( mu, sigma );

    plot(x, f);
end

% Squared−exp covariance function.
function k = covariance(x, y)
    k = exp( -0.5*( x - y )^2 );
end
```
Sampling from a GP

function simple_gp_sample

    % Choose a set of x locations.
    N = 100;
    x = linspace( -2, 2, N);

    % Specify the covariance between function values, depending on their location.
    for j = 1:N
        for k = 1:N
            sigma(j,k) = covariance( x(j), x(k) );
        end
    end

    % Specify that the prior mean of f is zero.
    mu = zeros(N, 1);

    % Sample from a multivariate Gaussian.
    f = mvnrnd( mu, sigma);

    plot(x, f);
end

% Squared-exp covariance function.
function k = covariance(x, y)
    k = exp( -0.5*( x - y )^2 );
end
Sampling from a GP

function simple_gp_sample

    % Choose a set of x locations.
    N = 100;
    x = linspace( -2, 2, N);

    % Specify the covariance between function values, depending on their location.
    for j = 1:N
        for k = 1:N
            sigma(j,k) = covariance( x(j), x(k) );
        end
    end

    % Specify that the prior mean of f is zero.
    mu = zeros(N, 1);

    % Sample from a multivariate Gaussian.
    f = mvnrnd( mu, sigma );

    plot(x, f);
end

% Squared-exp covariance function.
function k = covariance(x, y)
    k = exp( -0.5*( x - y )^2 );
end
Sampling from a GP

```matlab
function simple_gp_sample
    % Choose a set of x locations.
    N = 100;
    x = linspace( -2, 2, N);

    % Specify the covariance between function values, depending on their location.
    for j = 1:N
        for k = 1:N
            sigma(j,k) = covariance( x(j), x(k) );
        end
    end

    % Specify that the prior mean of f is zero.
    mu = zeros(N, 1);

    % Sample from a multivariate Gaussian.
    f = mvnrnd( mu, sigma );

    plot(x, f);
end

% Periodic covariance function.
function c = covariance(x, y)
    c = exp( -0.5*( sin(( x - y )*1.5).^2 ));
end
```
After conditioning on some data \((X, y)\),

\[
p(f(x_*)|X, y, \theta) = \frac{1}{Z} p(y|f, X, \theta)p(f|\theta)
\]
After conditioning on some data \((X, y)\),

\[
p(f(x_\star)|X, y, \theta) = \frac{1}{Z} p(y|f, X, \theta)p(f|\theta) \\
= \mathcal{N}(y|f, X, \theta)\mathcal{N}(f|\mu, K_\theta(X, X))
\]
After conditioning on some data \((X, y)\),

\[
p(f(x_*)|X, y, \theta) = \frac{1}{Z} p(y|f, X, \theta)p(f|\theta) \\
= \mathcal{N}(y|f, X, \theta)\mathcal{N}(f|\mu, K_{\theta}(X, X)) \\
= \mathcal{N}(f(x_*)|\mu = k(x_*, X)K^{-1}y, \\
\Sigma = k(x_*, x_*) - k(x_*, X)K^{-1}k(X, x_*))
\]
Conditional Posterior

\[ p(f(x_*)|X, y, \theta) = \mathcal{N}(f(x_*)|\mu = k(x_*, X)K^{-1}y, \\
\Sigma = k(x_*, x_*) - k(x_*, X)K^{-1}k(X, x_*)) \]
Conditional Posterior

\[ p(f(x_*)|X, y, \theta) = \mathcal{N}(f(x_*)|\mu = k(x_*, X)K^{-1}y, \Sigma = k(x_*, x_*) - k(x_*, X)K^{-1}k(X, x_*)). \]
Conditional Posterior

\[ p(f(x_*)|X, y, \theta) = \mathcal{N}(f(x_*)|\mu = k(x_*, X)K^{-1}y, \Sigma = k(x_*, x_*) - k(x_*, X)K^{-1}k(X, x_*)) \]
\[ p(f(x_*)|X, y, \theta) = \mathcal{N}(f(x_*)|\mu = k(x_*, X)K^{-1}y, \Sigma = k(x_*, x_*) - k(x_*, X)K^{-1}k(X, x_*)) \]
Conditional Posterior

\[
p(f(x_*)|X, y, \theta) = \mathcal{N}(f(x_*)|\mu = k(x_*, X)K^{-1}y, \\
\Sigma = k(x_*, x_*) - k(x_*, X)K^{-1}k(X, x_*))
\]
Problem with Bayesian models:

\[ p(f|X, y, \theta) = \frac{1}{Z} p(y|f, X, \theta)p(f|\theta) \]
Problem with Bayesian models:

\[
p(f|X, y, \theta) = \frac{1}{Z} p(y|f, X, \theta)p(f|\theta)
\]

\[
= \frac{p(y|f, X, \theta)p(f|\theta)}{\int p(y|f, X, \theta)p(f|\theta)df}
\]
Problem with Bayesian models:

\[
p(f|X, y, \theta) = \frac{1}{Z} p(y|f, X, \theta)p(f|\theta) = \frac{p(y|f, X, \theta)p(f|\theta)}{\int p(y|f, X, \theta)p(f|\theta)df}
\]

Can actually compute model evidence \( p(y|X, \theta) \), aka \( Z \):
Problem with Bayesian models:

\[
p(f|X, y, \theta) = \frac{1}{Z} p(y|f, X, \theta) p(f|\theta)
\]

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

Can actually compute model evidence \( p(y|X, \theta) \), aka \( Z \):

\[
\log p(y|X, \theta) = \log \int p(y|f, X, \theta) p(f|\theta) df
\]
Problem with Bayesian models:

\[
p(f|X, y, \theta) = \frac{1}{Z} p(y|f, X, \theta)p(f|\theta) \\
= \frac{p(y|f, X, \theta)p(f|\theta)}{\int p(y|f, X, \theta)p(f|\theta)df}
\]

Can actually compute model evidence \( p(y|X, \theta) \), aka \( Z \):

\[
\log p(y|X, \theta) = \log \int p(y|f, X, \theta)p(f|\theta)df \\
= -\frac{1}{2} y^T (K_\theta + \sigma^2 \mathbb{I})^{-1} y - \frac{1}{2} \log |K_\theta + \sigma^2 \mathbb{I}| - \frac{N}{2} \log(2\pi)
\]

- Data-fit
- Bayesian Occam’s Razor
Depending on kernel function, GP Regression is equivalent to:

- Linear Regression
- Polynomial Regression
- Splines
- Kalman Filters
- Generalized Additive Models

Can use gradients of model evidence to learn which model best explains the data; no need for cross-validation.
Depending on kernel function, GP Regression is equivalent to:

- Linear Regression
Choice of Kernel Function

Depending on kernel function, GP Regression is equivalent to:

- Linear Regression
- Polynomial Regression
Choice of Kernel Function

Depending on kernel function, GP Regression is equivalent to:

- Linear Regression
- Polynomial Regression
- Splines
Depending on kernel function, GP Regression is equivalent to:

- Linear Regression
- Polynomial Regression
- Splines
- Kalman Filters
Depending on kernel function, GP Regression is equivalent to:

- Linear Regression
- Polynomial Regression
- Splines
- Kalman Filters
- Generalized Additive Models
Depending on kernel function, GP Regression is equivalent to:

- Linear Regression
- Polynomial Regression
- Splines
- Kalman Filters
- Generalized Additive Models

Can use gradients of model evidence to learn which model best explains the data; no need for cross-validation.
Not just for 1-D continuous functions

▶ D-dimensional input
Not just for 1-D continuous functions

► D-dimensional input
► Functions over discrete domains
Not just for 1-D continuous functions

- D-dimensional input
- Functions over discrete domains
- Functions over strings, trees, trajectories
Not just for 1-D continuous functions

- D-dimensional input
- Functions over discrete domains
- Functions over strings, trees, trajectories
- Classification
Performance

[Blei et. al, 2011]
Limitations

- **Slow**: $O(N^3)$ means that $N < 3000$. 

- Recently some good $O(NM^2)$ approximations (FITC).

- Most commonly used kernels have fairly limited generalization abilities.

- Non-Gaussian noise requires approximate inference.

- Best choice if:
  - Data is small / expensive to gather.
  - You want to do anything besides point prediction.
Limitations

- Slow: $O(N^3)$ means that $N < 3000$.
  - Recently some good $O(NM^2)$ approximations (FITC).
Limitations

- **Slow**: $O(N^3)$ means that $N < 3000$.
  - Recently some good $O(NM^2)$ approximations (FITC).
- Most commonly used kernels have fairly limited generalization abilities.
Limitations

- Slow: $O(N^3)$ means that $N < 3000$.
  - Recently some good $O(NM^2)$ approximations (FITC).
- Most commonly used kernels have fairly limited generalization abilities.
- Non-Gaussian noise requires approximate inference.
Limitations

- Slow: $O(N^3)$ means that $N < 3000$.
  - Recently some good $O(NM^2)$ approximations (FITC).
- Most commonly used kernels have fairly limited generalization abilities.
- Non-Gaussian noise requires approximate inference.

Best choice if:
- Data is small / expensive to gather.
- You want to do anything besides point prediction.
Outline

Gaussian Process Regression
  Definition
  Properties

Additive Gaussian Processes
  Central Modeling Assumption
  Interpretability
  Related Work
  Results
Central modeling assumption:

\[ f_1(x_1) + f_2(x_2) = f_1(x_1) + f_2(x_2) \]
Central Dogma

Central modeling assumption:

\[ f_1(x_1) + f_2(x_2) = f_1(x_1) + f_2(x_2) \]

We hope our high-dimensional function can be written as a sum of orthogonal low-dimensional functions.
Central modeling assumption:

\[ f_1(x_1) + f_2(x_2) = f_1(x_1) + f_2(x_2) \]

We hope our high-dimensional function can be written as a sum of orthogonal low-dimensional functions.

it’s far easier to learn ten 1-dimensional functions than one 10-dimensional function!
Additivity in GPs

Easy to express additive property in a GP:

\[ k_1(x_1, x'_2) + k_2(x_2, x'_2) = k_1(x_1, x'_1) + k_2(x_2, x'_2) \]
Additivity in GPs

Easy to express additive property in a GP:

\[ k_1(x_1, x_2') + k_2(x_2, x_2') = k_1(x_1, x_1') + k_2(x_2, x_2') \]

\[ f_1(x_1) + f_2(x_2) \text{ draw from 1st order GP} \]
We can extend our prior to include more interaction terms:
We can extend our prior to include more interaction terms:

\[
f(x_1, x_2, x_3, x_4) =
\begin{align*}
f_1(x_1) &+ f_2(x_2) + f_3(x_3) + f_4(x_4) \\
+ f_{12}(x_1, x_2) &+ f_{13}(x_1, x_3) + f_{14}(x_1, x_4) + f_{23}(x_2, x_3) + f_{24}(x_2, x_4) + f_{34}(x_3, x_4) \\
+ f_{123}(x_1, x_2, x_3) &+ f_{124}(x_1, x_2, x_4) + f_{134}(x_1, x_3, x_4) + f_{234}(x_2, x_3, x_4) \\
+ f_{1234}(x_1, x_2, x_3, x_4)
\end{align*}
\]
We can extend our prior to include more interaction terms:

\[
\begin{align*}
  f(x_1, x_2, x_3, x_4) &= \\
  &= f_1(x_1) + f_2(x_2) + f_3(x_3) + f_4(x_4) \\
  &+ f_{12}(x_1, x_2) + f_{13}(x_1, x_3) + f_{14}(x_1, x_4) + f_{23}(x_2, x_3) + f_{24}(x_2, x_4) + f_{34}(x_3, x_4) \\
  &+ f_{123}(x_1, x_2, x_3) + f_{124}(x_1, x_2, x_4) + f_{134}(x_1, x_3, x_4) + f_{234}(x_2, x_3, x_4) \\
  &+ f_{1234}(x_1, x_2, x_3, x_4)
\end{align*}
\]

Corresponding GP model: assign each dimension \( i \in \{1 \ldots D\} \) a one-dimensional base kernel \( k_i(x_i, x'_i) \) Let \( z_i = k_i(x_i, x'_i) \)

\[
\begin{align*}
  k_{add_1}(x, x') &= z_1 + z_2 + z_3 + z_4 \\
  k_{add_2}(x, x') &= z_1z_2 + z_1z_3 + z_2z_3 + z_2z_4 + z_3z_4 \\
  k_{add_3}(x, x') &= z_1z_2z_3 + z_1z_2z_4 + z_1z_3z_4 + z_2z_3z_4 \\
  k_{add_4}(x, x') &= z_1z_2z_3z_4
\end{align*}
\]
In $D$ dimensions:

$$k_{add_1}(x, x') = \sigma_1^2 \sum_{i=1}^{D} k_i(x_i, x_i')$$
In $D$ dimensions:

$$k_{add_1}(x, x') = \sigma_1^2 \sum_{i=1}^{D} k_i(x_i, x'_i)$$

$$k_{add_2}(x, x') = \sigma_2^2 \sum_{i=1}^{D} \sum_{j=i+1}^{D} k_i(x_i, x'_i) k_j(x_j, x'_j)$$
In $D$ dimensions:

\[ k_{\text{add}_1}(x, x') = \sigma_1^2 \sum_{i=1}^{D} k_i(x_i, x'_i) \]

\[ k_{\text{add}_2}(x, x') = \sigma_2^2 \sum_{i=1}^{D} \sum_{j=i+1}^{D} k_i(x_i, x'_i)k_j(x_j, x'_j) \]

\[ k_{\text{add}_3}(x, x') = \sigma_3^2 \sum_{i=1}^{D} \sum_{j=i+1}^{D} \sum_{k=j+1}^{D} k_i(x_i, x'_i)k_j(x_j, x'_j)k_k(x_k, x'_k) \]

Full additive kernel is a sum of the additive kernels of all orders, weighted by the order variances $\sigma_1^2 ... \sigma_D^2$. 
In $D$ dimensions:

$$k_{add_1}(x, x') = \sigma_1^2 \sum_{i=1}^{D} k_i(x_i, x'_i)$$

$$k_{add_2}(x, x') = \sigma_2^2 \sum_{i=1}^{D} \sum_{j=i+1}^{D} k_i(x_i, x'_i) k_j(x_j, x'_j)$$

$$k_{add_3}(x, x') = \sigma_3^2 \sum_{i=1}^{D} \sum_{j=i+1}^{D} \sum_{k=j+1}^{D} k_i(x_i, x'_i) k_j(x_j, x'_j) k_k(x_k, x'_k)$$

$$k_{add_n}(x, x') = \sigma_n^2 \sum \prod_{1 \leq i_1 < i_2 < \ldots < i_n \leq D} k_{i_d}(x_{i_d}, x'_{i_d})$$
In $D$ dimensions:

$$k_{add_1}(x, x') = \sigma_1^2 \sum_{i=1}^{D} k_i(x_i, x'_i)$$

$$k_{add_2}(x, x') = \sigma_2^2 \sum_{i=1}^{D} \sum_{j=i+1}^{D} k_i(x_i, x'_i) k_j(x_j, x'_j)$$

$$k_{add_3}(x, x') = \sigma_3^2 \sum_{i=1}^{D} \sum_{j=i+1}^{D} \sum_{k=j+1}^{D} k_i(x_i, x'_i) k_j(x_j, x'_j) k_k(x_k, x'_k)$$

$$k_{add_n}(x, x') = \sigma_n^2 \sum_{1 \leq i_1 < i_2 < \ldots < i_n \leq D} \prod_{d=1}^{N} k_{i_d}(x_{i_d}, x'_{i_d})$$

$$k_{add_D}(x, x') = \sigma_D^2 \prod_{d=1}^{D} k_d(x_d, x'_d)$$

Full additive kernel is a sum of the additive kernels of all orders, weighted by the order variances $\sigma_1 \ldots \sigma_D$
Efficient Evaluation

The \( n \)th order additive kernel corresponds to the \( n \)th \textit{elementary symmetric polynomial}

\[
e_1(z_1, z_2, z_3, z_4) = z_1 + z_2 + z_3 + z_4
\]
\[
e_2(z_1, z_2, z_3, z_4) = z_1z_2 + z_1z_3 + z_1z_4 + z_2z_3 + z_2z_4 + z_3z_4
\]
\[
e_3(z_1, z_2, z_3, z_4) = z_1z_2z_3 + z_1z_2z_4 + z_1z_3z_4 + z_2z_3z_4
\]
\[
e_4(z_1, z_2, z_3, z_4) = z_1z_2z_3z_4
\]
Efficient Evaluation

The \( n \)th order additive kernel corresponds to the \( n \)th elementary symmetric polynomial

\[
e_1(z_1, z_2, z_3, z_4) = z_1 + z_2 + z_3 + z_4
\]
\[
e_2(z_1, z_2, z_3, z_4) = z_1z_2 + z_1z_3 + z_1z_4 + z_2z_3 + z_2z_4 + z_3z_4
\]
\[
e_3(z_1, z_2, z_3, z_4) = z_1z_2z_3 + z_1z_2z_4 + z_1z_3z_4 + z_2z_3z_4
\]
\[
e_4(z_1, z_2, z_3, z_4) = z_1z_2z_3z_4
\]

Newton-Girard formulae give efficient recursive form:

\[
e_n(z_1, \ldots, z_D) = \frac{1}{n} \sum_{k=1}^{n} (-1)^{(k-1)} e_{n-k}(z_1, \ldots, z_D) \sum_{i=1}^{D} z_i^k \quad (2.1)
\]
Interpretability

GP-GAM kernel

Squared-exp GP kernel

Additive GP kernel
## Interpretability

<table>
<thead>
<tr>
<th>Order</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
<th>6th</th>
<th>7th</th>
<th>8th</th>
<th>9th</th>
<th>10th</th>
</tr>
</thead>
<tbody>
<tr>
<td>pima</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.3</td>
<td>1.5</td>
<td><strong>96.4</strong></td>
<td>1.4</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>liver</td>
<td>0.0</td>
<td>0.2</td>
<td><strong>99.7</strong></td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>heart</td>
<td><strong>77.6</strong></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>22.0</td>
</tr>
<tr>
<td>concrete</td>
<td><strong>70.6</strong></td>
<td>13.3</td>
<td>13.8</td>
<td>2.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pumadyn</td>
<td>0.0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td><strong>99.5</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>servo</td>
<td><strong>58.7</strong></td>
<td>27.4</td>
<td>0.0</td>
<td>13.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>housing</td>
<td>0.1</td>
<td>0.6</td>
<td><strong>80.6</strong></td>
<td>1.4</td>
<td>1.8</td>
<td>0.8</td>
<td>0.7</td>
<td>0.8</td>
<td>0.6</td>
<td>12.7</td>
</tr>
</tbody>
</table>
Figure: Green points indicate the original data, blue points are data after the mean contribution from the other first-order terms has been subtracted. The black line is the posterior mean of a GP with only one term in its kernel.
Vapnik [1998] introduces additive kernel, calling it support vector ANOVA decomposition. Recommend choosing only one order, since must choose hypers by cross-validation.

Plate [1999] constructs additive GP using only first-order and $D$th order terms. Trades off interpretability with goodness of fit.

Wahba [1990] introduces smoothing-splines ANOVA, a weighted sum of low-D splines, each with an individual weight. In practice, only 1-D and 2-D splines are used.

Additive GPs use all orders of interaction, learn base kernels, are probabilistic.
Hierarchical Kernel Learning (Bach [2009])

- HKL can selects a hull of interaction terms.
- Must use a pre-determined weighting over orders.
- Uses cross-validation to fit all hypers.

Neither class of kernels contains the other.
Local Kernels

A GP prior with squared-exp or Matérn kernels say that either the function doesn’t change much at all, or that distant points can’t tell you much about your current position.

- Nice for consistency
- bad for generalization.

Squared-Exp kernel  Additive kernel
Non-Local Kernels

True Function & data locations

Squared-exp GP posterior mean

Additive GP posterior mean

Additive GP 1st-order functions
1st order interactions
\[ k_1 + k_2 + k_3 \]

2nd order interactions
\[ k_1 k_2 + k_2 k_3 + k_1 k_3 \]

3rd order interactions
\[ k_1 k_2 k_3 \]
(Squared-exp kernel)

All interactions
\[ k_1 + k_2 + \cdots + k_1 k_2 k_3 \]
(Additive kernel)
Results

Table: Regression Mean Squared Error

<table>
<thead>
<tr>
<th>Method</th>
<th>bach</th>
<th>concrete</th>
<th>pumadyn</th>
<th>servo</th>
<th>housing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>1.031</td>
<td>0.404</td>
<td>0.641</td>
<td>0.523</td>
<td>0.289</td>
</tr>
<tr>
<td>GP GAM</td>
<td>1.259</td>
<td>0.149</td>
<td>0.598</td>
<td>0.281</td>
<td>0.161</td>
</tr>
<tr>
<td>HKL</td>
<td>0.199</td>
<td>0.147</td>
<td>0.346</td>
<td>0.199</td>
<td>0.151</td>
</tr>
<tr>
<td>GP sq-exp</td>
<td>0.045</td>
<td>0.157</td>
<td>0.317</td>
<td>0.126</td>
<td>0.092</td>
</tr>
<tr>
<td>GP Additive</td>
<td>0.045</td>
<td>0.089</td>
<td>0.316</td>
<td>0.110</td>
<td>0.102</td>
</tr>
</tbody>
</table>

Table: Regression Negative Log Likelihood

<table>
<thead>
<tr>
<th>Method</th>
<th>bach</th>
<th>concrete</th>
<th>pumadyn</th>
<th>servo</th>
<th>housing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>2.430</td>
<td>1.403</td>
<td>1.881</td>
<td>1.678</td>
<td>1.052</td>
</tr>
<tr>
<td>GP GAM</td>
<td>1.708</td>
<td>0.467</td>
<td>1.195</td>
<td>0.800</td>
<td>0.457</td>
</tr>
<tr>
<td>GP sq-exp</td>
<td>−0.131</td>
<td>0.398</td>
<td>0.843</td>
<td>0.429</td>
<td>0.207</td>
</tr>
<tr>
<td>GP Additive</td>
<td>−0.131</td>
<td>0.114</td>
<td>0.841</td>
<td>0.309</td>
<td>0.194</td>
</tr>
</tbody>
</table>
Additive GPs generalize commonly used GPs and GAMs. Only penalty is time and $R$ extra hyperparameters.
Summary

- Additive GPs generalize commonly used GPs and GAMs. Only penalty is time and $R$ extra hyperparameters.
- Add a lot of tractable, interpretable structure to your model.
Additive GPs generalize commonly used GPs and GAMs. Only penalty is time and $R$ extra hyperparameters.

- Add a lot of tractable, interpretable structure to your model.
- Allows better generalization if the data supports it.
A puzzle

An experiment Carl did:

- Draw an 8-dimensional function from a GP with a sq-exp kernel

2D Sq-exp kernel → Draw from a GP with a 2D Sq-exp kernel
A puzzle

An experiment Carl did:

- Draw an 8-dimensional function from a GP with a sq-exp kernel

2D Sq-exp kernel

→

Draw from a GP with a 2D Sq-exp kernel

- Draw train and test points from a Gaussian centered at 0
A puzzle

An experiment Carl did:

- Draw an 8-dimensional function from a GP with a sq-exp kernel

2D Sq-exp kernel \[ \rightarrow \] Draw from a GP with a 2D Sq-exp kernel

- Draw train and test points from a Gaussian centered at 0
- Predict test points from training points
A puzzle

An experiment Carl did:

- Draw an 8-dimensional function from a GP with a sq-exp kernel

2D Sq-exp kernel → Draw from a GP with a 2D Sq-exp kernel

- Draw train and test points from a Gaussian centered at 0
- Predict test points from training points
- How many training points needed to learn?
A puzzle

An experiment Carl did:

- Draw an 8-dimensional function from a GP with a sq-exp kernel

2D Sq-exp kernel  \rightarrow  Draw from a GP with a 2D Sq-exp kernel

- Draw train and test points from a Gaussian centered at 0
- Predict test points from training points
- How many training points needed to learn?

Learning a high-dimensional function from this model class requires exponentially many training points.
A puzzle

An experiment Carl did:

- Draw an 8-dimensional function from a GP with a sq-exp kernel

![2D Sq-exp kernel](image1)

![Draw from a GP with a 2D Sq-exp kernel](image2)

- Draw train and test points from a Gaussian centered at 0
- Predict test points from training points
- How many training points needed to learn?

Learning a high-dimensional function from this model class requires exponentially many training points.

How is it that sq-exp GP regression actually works on high-dimensional functions?
The end!
The end!
Ideas and criticism welcome.

