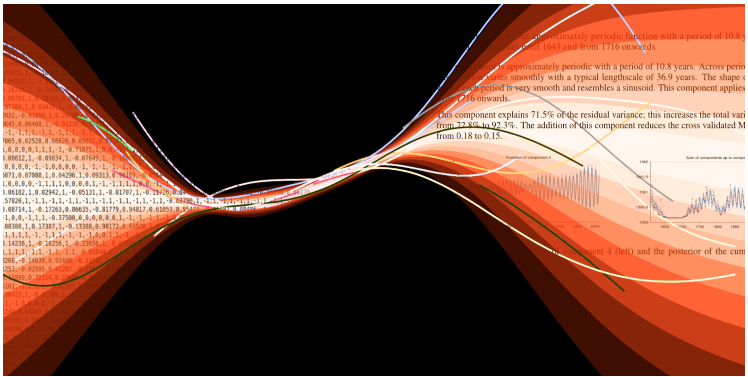
Gaussian Processes for Regression

Concept of Gaussian Processes
Gaussian Processes for Control
Application to fMRI

Joachim M. Buhmann

October 21, 2019

Gaussian Processes



Teaser image: (c) 2014 by James Robert Lloyd, David Duvenaud and Roger Grosse

Bayesian linear regression

Motivation: To understand Gaussian processes, let us begin by considering how we can turn the conventional multiple linear regression model into a Bayesian regression model. Recall that in regression we are trying to predict an output variable $Y \in \mathbb{R}$ from an input vector $X \in \mathbb{K}$, e.g. $\mathbb{K} \subset \mathbb{R}^d$.

Given an input vector $X = (X_1, \ldots, X_d)^\mathsf{T}$ and model parameters $\beta \in \mathbb{R}^d$, a multiple linear regression model explains the output (or response) variable as

$$Y = X^\mathsf{T}\beta + \epsilon \quad \text{with} \quad \epsilon \sim \mathcal{N}(\epsilon|0, \sigma^2),$$

or equivalently:

$$p\left(Y|X, \beta, \sigma\right) = \mathcal{N}\left(Y|X^\mathsf{T}\beta, \sigma^2\right) \propto \exp\left(-\tfrac{1}{2\sigma^2}(Y - X^\mathsf{T}\beta)^2\right)$$

Bayesian linear regression

Bayesian linear regression extends multiple linear regression by defining a prior over the regression coefficients, for example (ridge regression):

$$p(\beta|\Lambda) = \mathcal{N}_d\left(\beta|\mathbf{0}, \Lambda^{-1}\right) \propto \exp\left(-\tfrac{1}{2}\beta^\mathsf{T}\Lambda\beta\right)$$

Model inversion: given observed data ($\mathbf{X} \in \mathbb{R}^{n\times d}, \mathbf{y} \in \mathbb{R}^n$) we can now use Bayes' theorem to obtain the posterior distribution over coefficients:

$$\begin{aligned} p(\beta|\mathbf{X}, \mathbf{y}, \Lambda) &= \mathcal{N}(\beta|\mu_\beta, \Sigma_\beta) \\ \mu_\beta &= \left(\mathbf{X}^\mathsf{T}\mathbf{X} + \sigma^2\Lambda\right)^{-1}\mathbf{X}^\mathsf{T}\mathbf{y} \\ \Sigma_\beta &= \sigma^2\left(\mathbf{X}^\mathsf{T}\mathbf{X} + \sigma^2\Lambda\right)^{-1} \end{aligned}$$

Note: Bayesian linear regression with a Gaussian prior over coefficients is equivalent to ridge regression for $\Lambda = \lambda\mathbb{I}_d$ and $\sigma = 1$.

From Bayesian linear regression to Gaussian processes

We might ask:

Given a Gaussian prior over coefficients β and a given noise variance σ, what is the joint distribution of all outputs $\mathbf{y}$? Since we have modelled the individual outputs Y as conditionally independent (given inputs and coefficients), the vector of all outputs, $\mathbf{y}$, is given by:

$$\mathbf{y} = \mathbf{X}\beta + \epsilon, \qquad \epsilon \sim \mathcal{N}_n\left(\epsilon | 0, \sigma \mathbb{I}_n\right) \quad \text{(matrix notation!)}$$

Since the outputs $\mathbf{y}$ are a linear combination of normally distributed random variables β, they are jointly Gaussian themselves.

Moments of Bayesian linear regression

From previous slide: $\mathbf{y} = \mathbf{X}\beta + \epsilon, \quad \epsilon \sim \mathcal{N}(\epsilon|0, \sigma\mathbb{I}_n)$

The moments of the joint Gaussian in β, ϵ are:

$$\begin{aligned}
\mathbb{E}_{\beta,\epsilon}[\mathbf{y}] &= \mathbf{X}\mathbb{E}_\beta[\beta] + \mathbb{E}_\epsilon[\epsilon] = \mathbf{X}\mathbf{0} + \mathbf{0} = \mathbf{0} \\
\mathbb{C}\text{ov}[\mathbf{y}] &= \mathbb{E}_{\beta,\epsilon}\left[(\mathbf{X}\beta + \epsilon)(\mathbf{X}\beta + \epsilon)^\mathsf{T}\right] \\
&= \mathbf{X}\mathbb{E}_\beta\left[\beta\beta^\mathsf{T}\right]\mathbf{X}^\mathsf{T} + \mathbf{X}\mathbb{E}_\beta[\beta]\mathbb{E}_\epsilon\left[\epsilon^\mathsf{T}\right] + \mathbb{E}_\epsilon[\epsilon]\mathbb{E}_\beta\left[\beta^\mathsf{T}\right]\mathbf{X}^\mathsf{T} + \mathbb{E}_\epsilon\left[\epsilon\epsilon^\mathsf{T}\right] \\
&= \mathbf{X}\Lambda^{-1}\mathbf{X}^\mathsf{T} + \sigma^2\mathbb{I}_n
\end{aligned}$$

Special case: diagonal covariance for β

$$\mathbb{C}\text{ov}[\mathbf{y}] = \lambda^{-1}\left(\mathbf{X}\mathbf{X}^\mathsf{T} + \lambda\sigma^2\mathbb{I}_n\right) \quad \text{for} \quad \Lambda = \lambda\mathbb{I}_d$$

Gaussian processes

Moments of joint Gaussian: $\mathbb{E}[\mathbf{y}] = \mathbf{0}, \mathbb{C}\text{ov}[\mathbf{y}] = \mathbf{X}\Lambda^{-1}\mathbf{X}^\mathsf{T} + \sigma^2\mathbb{I}_n$

We can rewrite the joint distribution over $\mathbf{y}$ as follows:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \sim \mathcal{N}\left(\mathbf{y} \,\middle|\, \mathbf{0}, \begin{bmatrix} k_{1,1}+\sigma^2 & k_{1,2} & \dots & k_{1,n} \\ k_{2,1} & k_{2,2}+\sigma^2 & \dots & k_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ k_{n,1} & k_{n,2} & \dots & k_{n,n}+\sigma^2 \end{bmatrix}\right)$$

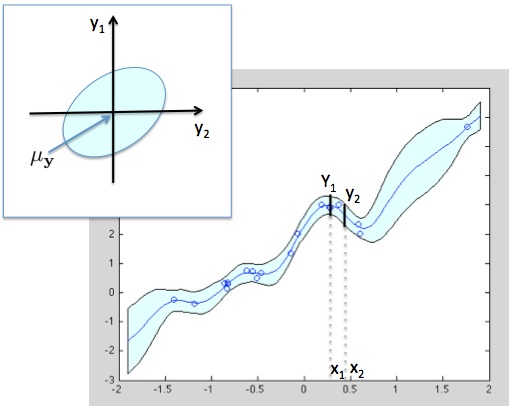
where $k_{i,j} = k(x_i, x_j) := x_i^\mathsf{T}\Lambda^{-1}x_j$ is a so-called kernel function.

Gaussian Processes as "kernelized linear regression"

This formulation of Bayesian regression with general $k(x_i, x_j)$ is no longer necessarily linear and is known as Gaussian process regression. Its power and flexibility derive from the fact that any other kernel function could be used instead of $x_i^\mathsf{T}\Lambda^{-1}x_j$. (Reminder: Kernels have to be validated!)

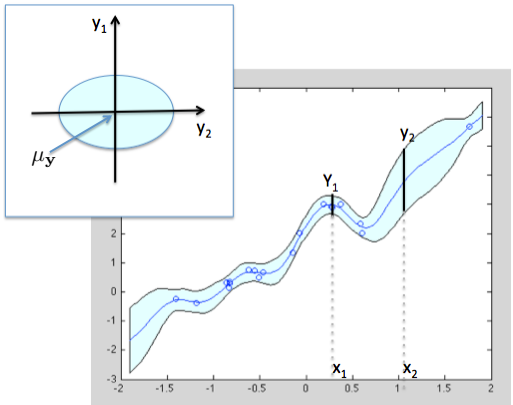
Gaussian processes: large covariance

The kernel function on the previous slide expresses that the outputs of two points whose inputs are 'similar' to each other have a high covariance.



Gaussian processes: low covariance

By contrast, the outputs of points with “dissimilar” inputs have a low covariance.
The level set of the joint distribution yields an axis aligned ellipse.

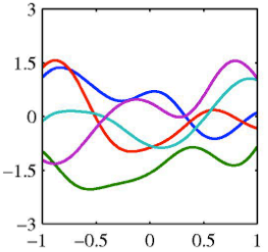


Gaussian processes

Kernel functions specify the degree of similarity between any two data points.

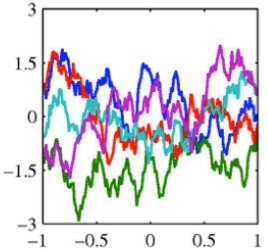
The kernel functions encode our assumptions about the function which we wish to learn, e.g. its smoothness. Different kernel functions can be used to obtain very different models. This adaptivity is in contrast to conventional regression, where we typically fix the model class and only allow for variation in the coefficients.

Gaussian kernel



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

Exponential kernel



$$k(x_i, x_j) = \exp\left(-\theta|x_i - x_j|\right)$$

Recall: kernel properties

Large freedom in selecting the optimal kernel function for some specific application.

Kernel functions $k(x, x')$ must satisfy (cf. properties of covariance matrices):

1. Symmetry: $k(x, x') = k(x', x)$
2. Positive semi-definiteness (continuous case):

$$\int_{\Omega} k(x, x') f(x) f(x') \, \mathrm{d}x \, \mathrm{d}x' \geq 0 \quad \forall f \in L_2, \; \Omega \subset \mathbb{R}^d$$

Kernels represent scalar products in corresponding Euclidean spaces. Such spaces could be very high dimensional (even infinite dimensional) but the evaluation of the kernel yields the correct result for the scalar product without constructing explicit vector representations.

Recall: Gram matrix

Corresponding condition for a kernel function to be positive semi-definite (discrete case):

For any $n \in \mathbb{N}$, any set $S = \{x_1, \ldots, x_n\}$, the kernel (Gram) matrix

$$\mathbf{K} = \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{pmatrix}$$

must be positive semi-definite.

Remember: $\mathbf{K}$ is positive semi-definite $\Leftrightarrow$ $x^{\mathsf{T}}\mathbf{K}x \geq 0$ for all x

Recall: Examples of kernel functions

Some classical kernels on $\mathbb{R}^d$:

Linear kernel:	$k(x, x') = x^\mathsf{T} x'$
Polynomial kernel:	$k(x, x') = (x^\mathsf{T} x' + 1)^p$, for $p \in \mathbb{N}$
Gaussian (RBF) kernel:	$k(x, x') = \exp\left(-\|x - x'\|_2^2 / h^2\right)$
Sigmoid (tanh) kernel:	$k(x, x') = \tanh \kappa x^\mathsf{T} x' - b$

Different kernels have different invariance properties! For example, invariance to rotation or translation.

Recall: Kernel engineering by composition

Given two kernel functions $k_1(x, x')$ and $k_2(x, x')$ defined on the same data space, new kernel functions $k(x, x')$ can be constructed by the following rules:

Addition:	$k(x, x') = k_1(x, x') + k_2(x, x')$
Multiplication:	$k(x, x') = k_1(x, x') \cdot k_2(x, x')$
Scaling:	$k(x, x') = c \cdot k_1(x, x')$ for $c > 0$
Composition:	$k(x, x') = f(k_1(x, x'))$ where f is a polynomial with positive coefficients or the exponential function.

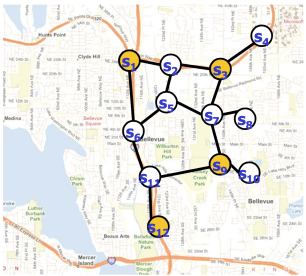
Kernel engineering is important in many domains!

It allows us to flexibly model machine learning solutions for various data types like strings, graphs, trees, lists, etc., when the algorithm depends only on scalar products.

GPs: Kernels beyond $\mathbb{R}^D$

Kernels can be defined on a variety of objects:

- Sequence kernels
- Graph kernels
- Diffusion kernels
- Kernels on probability distributions (e.g. $k(x, x') = \sum_{c \in \mathcal{C}} p(c|x)p(c|x') \in [0, 1]$)
- ...



For example, we can measure similarity among nodes in a graph via diffusion kernels.
(Imagine a Markov process on a graph with transition probabilities given by "normalized" edge weights.)

Prediction by Gaussian processes

The predictive density $p\left(y_{n+1}|x_{n+1}, \mathbf{X}, \mathbf{y}\right)$

of the output y_{n+1} for a new data point x_{n+1} can be obtained analytically. For this, we derive the joint distribution

$$p\left(\left[\begin{array}{c}\mathbf{y}\\ y_{n+1}\end{array}\right]\middle| x_{n+1}, \mathbf{X}, \sigma\right) = \mathcal{N}\left(\left[\begin{array}{c}\mathbf{y}\\ y_{n+1}\end{array}\right]\middle| \mathbf{0}, \left[\begin{array}{cc}\mathbf{C}_n & \mathbf{k}\\ \mathbf{k}^{\mathsf{T}} & c\end{array}\right]\right),$$

$$\begin{aligned}\text{where}\quad \mathbf{C}_n &= \mathbf{K} + \sigma^2\mathbf{I},\\ c &= k\left(x_{n+1}, x_{n+1}\right) + \sigma^2,\\ \mathbf{k} &= k\left(x_{n+1}, \mathbf{X}\right),\\ \text{and}\quad \mathbf{K} &= k\left(\mathbf{X}, \mathbf{X}\right).\end{aligned}$$

Note that the joint distribution still has a vanishing mean since $\mathbb{E}_\beta[\beta] = \mathbf{0}$ in $\mathbf{y} = \mathbf{X}\beta + \epsilon$ (no preference of specific non-zero $\mathbf{y}$ values).

Gaussian processes

Prediction by Gaussian processes

Reminder: Conditional Gaussian Distributions

$$p\left(\begin{bmatrix}\mathbf{a_1}\\ \mathbf{a_2}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{a_1}\\ \mathbf{a_2}\end{bmatrix}\middle|\begin{bmatrix}\mathbf{u_1}\\ \mathbf{u_2}\end{bmatrix}, \begin{bmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{bmatrix}\right)$$

with

$$\begin{array}{llll}\mathbf{a_1}, \mathbf{u_1} \in \mathbb{R}^e & \Sigma_{11} \in \mathbb{R}^{e\times e} \text{ psd} & \Sigma_{12} \in \mathbb{R}^{e\times f} \text{ psd}\\ \mathbf{a_2}, \mathbf{u_2} \in \mathbb{R}^d & \Sigma_{22} \in \mathbb{R}^{f\times f} \text{ psd} & \Sigma_{21} \in \mathbb{R}^{f\times e} \text{ psd}\end{array}$$

Then,

$$p(\mathbf{a_2} \mid \mathbf{a_1} = \mathbf{z}) = \mathcal{N}\left(\mathbf{a_2} \mid \mathbf{u}_2 + \Sigma_{21}\Sigma_{11}^{-1}(\mathbf{z} - \mathbf{u_1}), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}\right).$$

Gaussian processes

Prediction by Gaussian processes

The predictive density $p\left(y_{n+1}|x_{n+1},\mathbf{X},\mathbf{y}\right)$ becomes

$$p\left(y_{n+1}|x_{n+1},\mathbf{X},\mathbf{y}\right)=\frac{p\left(\left[\mathbf{y}^{\mathsf{T}},y_{n+1}\right]^{\mathsf{T}}|x_{n+1},\mathbf{X},\right)}{\int_{y_{n+1}}p\left(\left[\mathbf{y}^{\mathsf{T}},y_{n+1}\right]^{\mathsf{T}}|x_{n+1},\mathbf{X}\right)\mathrm{d}y_{n+1}},$$

where

$$\begin{aligned}p\left(y_{n+1}|x_{n+1},\mathbf{X},\mathbf{y}\right) &= \mathcal{N}\left(y_{n+1}|\,\mu_{y_{n+1}},\sigma^2_{y_{n+1}}\right)\\ \mu_{y_{n+1}} &= \mathbf{k}^{\mathsf{T}}\mathbf{C}_n^{-1}\mathbf{y}\;=\;\mathbf{k}^{\mathsf{T}}(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{y}\\ \sigma^2_{y_{n+1}} &= c-\mathbf{k}^{\mathsf{T}}\mathbf{C}_n^{-1}\mathbf{k}\end{aligned}$$

GPs: Pseudo-Code for Prediction

Algorithm Prediction with Gaussian Processes

Require: n observed data ($\mathbf{X} = (x_1, \ldots, x_n)^\mathsf{T} \in \mathbb{R}^{n \times d}$, $\mathbf{y} \in \mathbb{R}^n$), kernel function k, noise variance σ^2, new data point $x_{n+1} \in \mathbb{R}^d$

$\mathbf{K} \leftarrow (k(x_i, x_j))_{1 \leq i,j \leq n}$ // Compute kernel matrix

$\mathbf{k} \leftarrow (k(x_{n+1}, x_i))_{1 \leq i \leq n}$ // Similarity of new data point and observed data

$\mu_{y_{n+1}} \leftarrow \mathbf{k}^\mathsf{T}(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}$ // Mean of predictive distribution

$\sigma^2_{y_{n+1}} \leftarrow k(x_{n+1}, x_{n+1}) - \mathbf{k}^\mathsf{T}(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}$ // Variance of predictive distribution

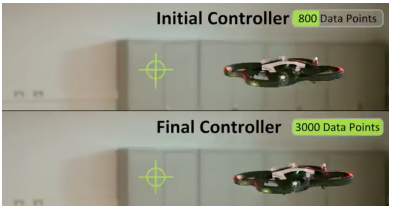
return $\mathcal{N}(y_{n+1} | \mu_{y_{n+1}}, \sigma^2_{y_{n+1}})$ // Return predictive distribution

The prediction algorithm returns a distribution function. The prediction at x_{n+1} yields a mean value $\mu_{y_{n+1}}$ and a variance $\sigma^2_{y_{n+1}}$. Furthermore, samples y_{n+1} can be drawn from this distribution.

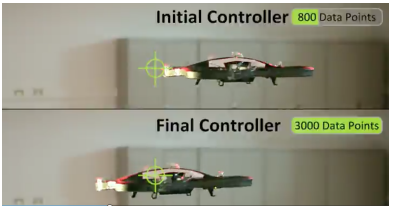
Controller Optimization for Robust Control

Machine Learning in Control Systems

Machine learning techniques are becoming more and more important for enabling computers to control complex and stochastic systems and predict the outcomes of such systems.



$\Rightarrow$



Quadrotor should reach final position fast and safely

https://www.youtube.com/watch?v=7ZkZlxXHgTY,
work by Felix Berkenkamp, Angela P. Schoellig, and Andreas Krause

Gaussian processes for Control (1)

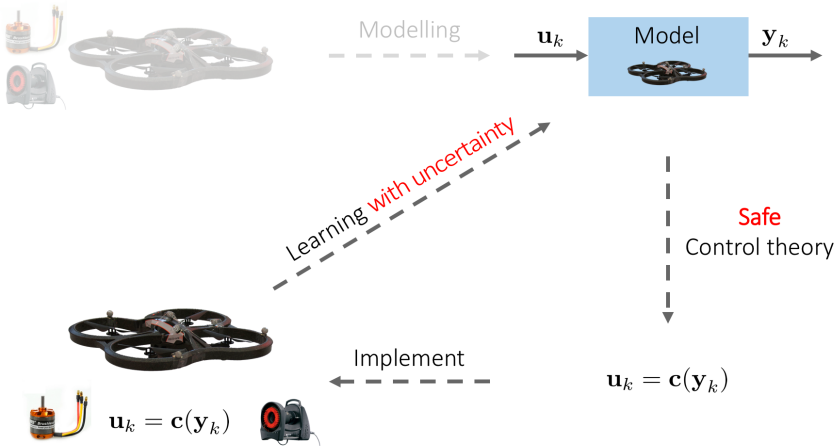
A Fundamental problem when designing controllers for dynamic systems is the tuning of the controller parameters. Besides pure performance, robustness is an important issue.

The classical approach is to use a model of the system to design an initial controller; parameters are then tuned manually to achieve best performance.

An alternative approach uses methods from machine learning to optimize performance, e.g., Bayesian optimization.

Safety-critical system failures may happen because these methods evaluate different controller parameters.

Gaussian processes for Control (2)

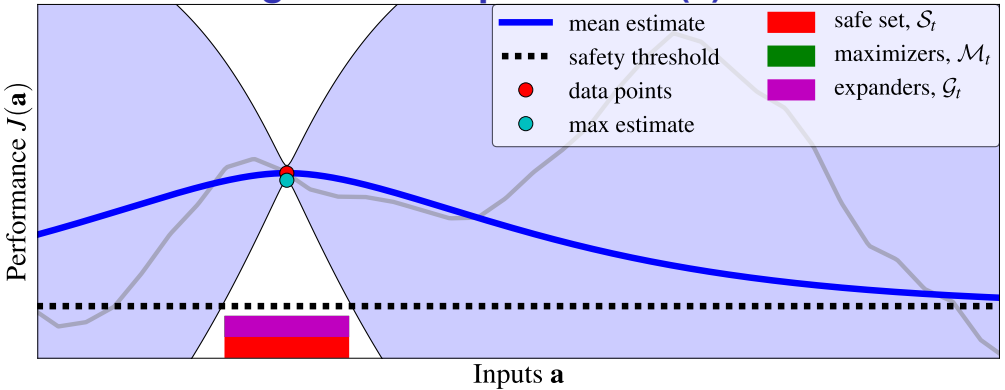


Safe Control using Gaussian processes (1)

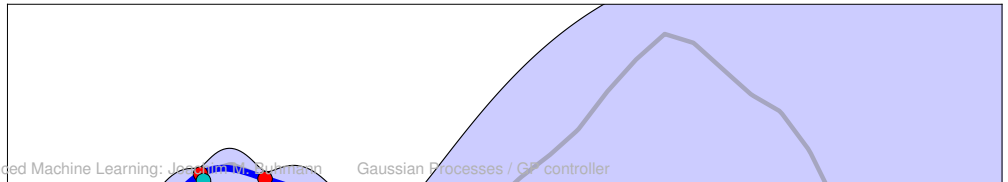
Safe optimization

Overcome safety-critical system failures by using a specialized optimization algorithm for automatic controller parameter tuning. This algorithm models the underlying performance measure as a GP and only explores new controller parameters whose performance lies above a safe performance threshold with high probability.

Safe Control using Gaussian processes (2)



Initial, safe parameters.

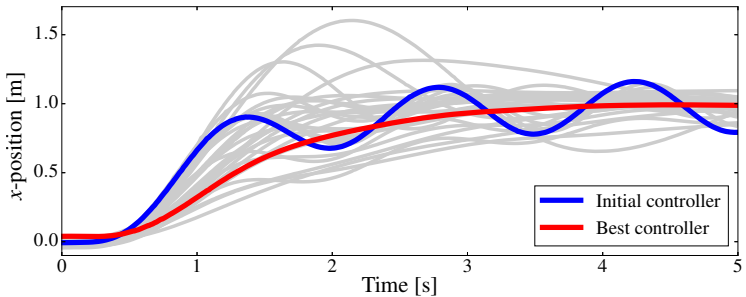


Learning to Fly a Quadrotor

Goal and Safety-constranits

Quickly identify the optimal parameters, with maximum performance, on real, closed-loop system only evaluating controllers above a safe performance threshold.

For quadrotor performance measured in reference position change.



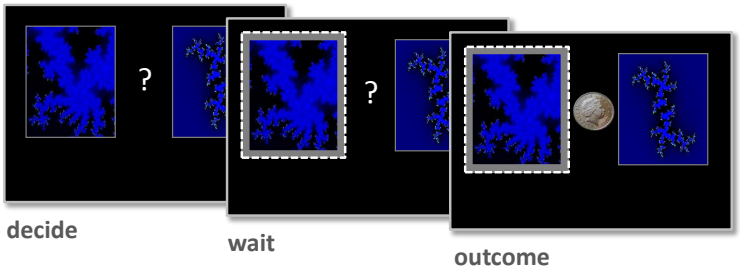
In action: ▸ Link

Neuroimaging to discover brain function

Machine Learning in Neuroimaging

A rich source of machine-learning challenges can be found in the field of computational neuroimaging which aims to understand structure-function relationships in the human brain.

MRI scanner: one volume of 50,000 voxels every 3 seconds



Gaussian processes and neuroimaging

In this experiment, participants were asked to solve a series of simple decision-making problems while undergoing functional magnetic resonance imaging (fMRI). Decisions were indicated by pressing a button with the left or right index finger.

We hypothesized that brain activity at the time of the button press would allow us to decode, on a trial-by-trial basis, which decision had been made. We tested this hypothesis using a Gaussian process model for classification.

The analysis is challenging because each brain scan (i.e., the input variable X) contains about 50,000 features (voxels), while data acquisition is limited to no more than $N = 160$ trials per subject.

Classification using Gaussian processes

Classification accuracy

Using leave-one-out cross-validation, the GP model achieves a classification accuracy of 61%, which is significantly above chance $(p < 0.01)$, confirming the original hypothesis.

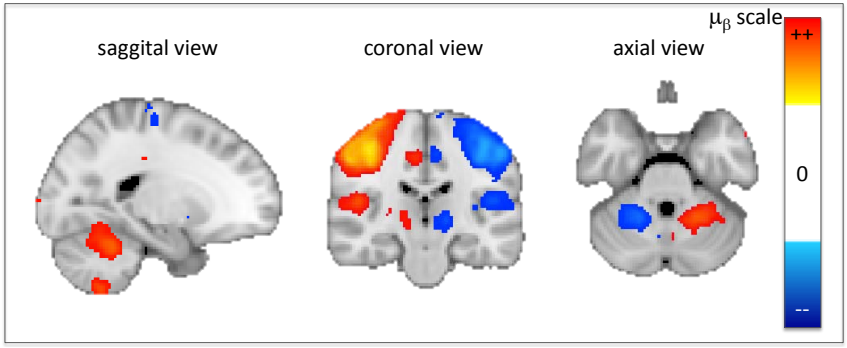
Interpretability

In addition to prediction performance, we are interested in the model coefficients

$$\beta = \mathbf{X}^{\mathsf{T}} C^{-1} \mathbf{y} \; .$$

Using all coefficients that are significantly different from zero, we can obtain a map of brain regions whose activity contains information about what decision has just been made.

Mapping discriminative voxels across the brain



In this map, red/yellow voxels were found to be indicative of trials in which a subject decided to press the right button. In contrast, blue voxels represent voxels whose activity was indicative of left button presses. The map shows how the left hemisphere becomes active when the right body half is being used, and vice versa.

Maps like the one shown above have proven very useful in understanding structure-function relationships in the human brain.

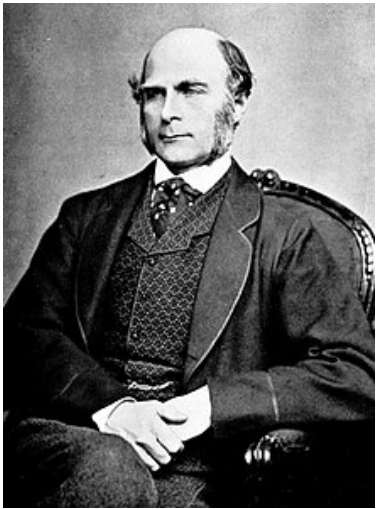
Model averaging is common practice

- Previous: Gaussian process motivated by Bayesian linear regression.
- Seldom: take MAP estimator in Bayesian setting.
- **Bayesian** approach: average models with different parameters (weighted according to prior).
- Cross validation: Take average over models trained on different folds.
- Winners of most Machine Learning competitions (e.g. on Kaggle): **ensembles** (weighted averages of models).

Not only true for machines...

Wisdom of crowds

- Sir Francis Galton asked 787 people in 1906 to estimate the weight of an ox.
- None got the right answer.
- Yet, the average prediction was precise (correct weight: 1198 pounds, avg. estimate: 1197 pounds).



Next: Why model averaging is a good idea.

Combining Regressors - Bias

Set of estimators: $\hat{f}_1(x), \hat{f}_2(x), \ldots, \hat{f}_B(x)$

Simple average: $\hat{f}(x) = \frac{1}{B}\sum_{i=1}^{B} \hat{f}_i(x)$

Bias

$$\begin{aligned} \text{bias}[\hat{f}(x)] &= \mathbb{E}_D \hat{f}(x) - \mathbb{E}(Y|x) = \frac{1}{B}\sum_{i=1}^{B} \mathbb{E}_D \hat{f}_i(x) - \mathbb{E}(Y|x) \\ &= \frac{1}{B}\sum_{i=1}^{B} \left(\mathbb{E}_D \hat{f}_i(x) - \mathbb{E}(Y|x) \right) = \frac{1}{B}\sum_{i=1}^{B} \text{bias}[\hat{f}_i(x)] \end{aligned}$$

Unbiased estimators remain unbiased after averaging

Combining Regressors - Variance I

$$
\begin{aligned}
\mathbb{V}[\hat{f}(x)] &= \mathbb{E}_D\left(\hat{f}(X) - \mathbb{E}_D\hat{f}(X)\right)^2 \\
&= \mathbb{E}_D\left(\frac{1}{B}\sum_{i=1}^{B}\hat{f}_i(x) - \frac{1}{B}\sum_{i=1}^{B}\mathbb{E}_D\hat{f}_i(x)\right)^2 \\
&= \mathbb{E}_D\left(\frac{1}{B}\sum_{i=1}^{B}\left(\hat{f}_i(x) - \mathbb{E}_D\hat{f}_i(x)\right)\right)^2 \\
&= \frac{1}{B^2}\sum_{i=1}^{B}\mathbb{V}_D\left[\hat{f}_i(x)\right] + \frac{1}{B^2}\sum\sum_{i\neq j}\mathbb{C}\text{ov}\left[\hat{f}_i(x),\hat{f}_j(x)\right]
\end{aligned}
$$

Combining Regressors - Variance II

Recall

$$\mathbb{V}[\hat{f}(x)] = \frac{1}{B^2}\sum_{i=1}^{B}\mathbb{V}\left[\hat{f}_i(x)\right] + \frac{1}{B^2}\sum_{i\neq j}\sum\mathbb{C}\text{ov}\left[\hat{f}_i(x),\hat{f}_j(x)\right]$$

Assume:

Covariances small: $\mathbb{C}\text{ov}\left[\hat{f}_i(x),\hat{f}_j(x)\right] \approx 0$

Variances similar: $\mathbb{V}\left[\hat{f}_i(x)\right] \approx \sigma^2$

Then

$$\mathbb{V}[\hat{f}(x)] \approx \frac{\sigma^2}{B}$$

Reduction by a factor of $1/B$ - main effect if bias remains unchanged