

Gaussian Processes and Bayesian Optimization

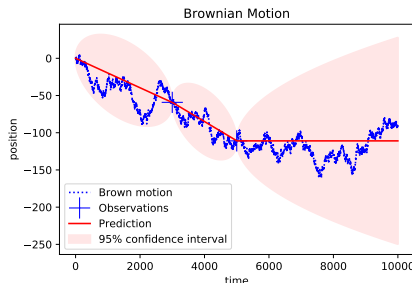
- Bayesian Optimization is used when
 - We are solving: $x^* = \arg \min_x f(x)$
 - $f(x)$ is a black box function
 - f is expensive to evaluate
 - the evaluations may be noisy.
- If any condition is not true, a better algorithm exists.
- We search the point $\mathbf{x}$ to observe.
- scikit-optimize = skopt Python package
- we minimize $\mathbf{y}$ and search the maximal probability of improvement
- 'the chance to improve' is expressed by the **Expected improvement** (EI)

Bayesian Optimization Algorithm

- Evaluate $\mathbf{y}$ on X , let $\mathbf{y} = y(X)$ and calculate conditional means and covariances
- repeat forever
 - $x^{new} = \operatorname{argmax}_x EI(x)$ add x into X
 - Evaluate $\mathbf{y} = y(x)$ and add y to $\mathbf{y}$.
 - re-estimate the **Gaussian process (the parameters of the covariance (kernel) function)**.

Gaussian Processes

- An infinite (continuous) number of Gaussian variables
- any variable x follows $N(\mu = f(x), \Sigma_{x|observed})$
- since we have only a finite number of observations which means a finite number of variables
 - we can marginalize unobserved variables out (the integral is 1, we multiply by 1, we just remove),
- we can predict at any x , continuously.



Gaussian Processes

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006

Definition (Gaussian Process)

A Gaussian process is a set of random variables where any finite subset follows multivariate Gaussian distribution.

We define the mean $m(\mathbf{x})$ and the symmetric positive semidefinite covariance function $k(\mathbf{x}, \mathbf{x}^l)$:

$$\begin{aligned}m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\k(\mathbf{x}, \mathbf{x}^l) &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}^l) - m(\mathbf{x}^l))]\end{aligned}$$

a Gaussian process is

$$f(\mathbf{x}) = \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}^l)).$$

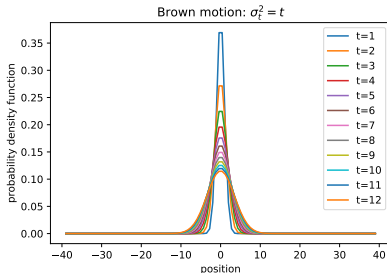
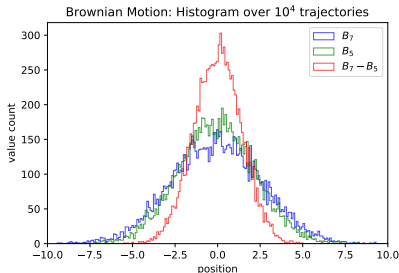
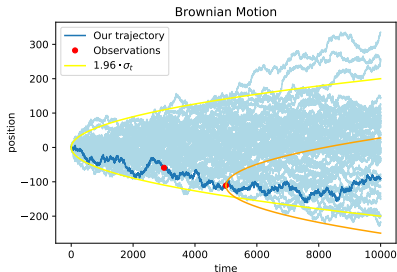
We assume $m(\mathbf{x}) = \mathbf{0}$ it simplifies the formulas.

Brownian Motion (Wiener Process)

<https://www.coursera.org/lecture/stochasticprocesses/week-4-6-two-definitions-of-a-brownian-motion-THRqL>

Definition (Brownian motion 1)

- $B_0 = 0$ for sure
- stationary and independent increments
- $B_s - B_t \sim N(0, s - t)$



Definition (Brownian motion 1)

- $B_0 = 0$ almost surely
- B_t stationary and independent increments
- $B_s - B_t \sim N(0, s - t)$

Definition (Brownian motion 2)

Gaussian process with

- $m = 0$ and
- $k(x, x') = \min(x, x')$.

Positive semidefinite:

- $\min(t, s) = \int_0^\infty f_t(x) f_s(x) dx$
- $f_t(x) f_s(x) = 1$ iff $x \in [0, t] \& x \in [0, s]$

Lemma (2 $\Rightarrow$ 1)

- $K(0, 0) = \min(0, 0) = 0$
- The process has variance 0 at $t = 0$ and $m(0) = 0$.
- covariance is linear in both arguments, $s \geq t$

$$\begin{aligned} \text{cov}(B_s - B_t, B_s - B_t) &= \text{cov}(B_s, B_s) - \text{cov}(B_t, B_s) - \text{cov}(B_s, B_t) + \text{cov}(B_t, B_t) \\ &= s - 2t + t = s - t \end{aligned}$$

- increments, $s \geq t \geq b \geq a$ # independence skipped, from Gaussian vectors

$$\begin{aligned} \text{cov}(B_b - B_a, B_s - B_t) &= \text{cov}(B_b, B_s) - \text{cov}(B_a, B_s) - \text{cov}(B_b, B_t) + \text{cov}(B_a, B_t) \\ &= b - a - b + a = 0. \end{aligned}$$

Normal Distribution

Definition (Brownian motion 2)

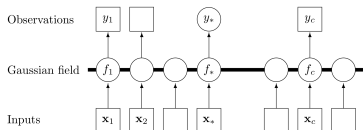
Gaussian process with

- $m = 0$ and
- $k(x, x') = \min(x, x')$.
- The covariance on $\mathbf{y}$ is defined by the covariance on the inputs $\mathbf{x}$.
- the covariance defines also the distribution on functions f :

$$\mathbf{f}_* \sim N(\mathbf{0}, K(X_*, X_*)).$$

- Without noise, we observe $\mathbf{y}$ and we want to predict $\mathbf{f}_*$:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim N\left(\mathbf{0}, \begin{bmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right)$$



$$X = [3, 5]$$

$$\mathbf{y}^T = [-5, -11]$$

$$K(x_*, X) = [\min(x_*, a) \text{ for } a \text{ in } X] \quad K(X, X) = \begin{bmatrix} \min(3, 3) & \min(3, 5) \\ \min(3, 5) & \min(5, 5) \end{bmatrix} = \begin{bmatrix} 3 & 3 \\ 3 & 5 \end{bmatrix}$$

Prediction

- noisy-free observations $y = f(\mathbf{x})$

$$\text{cov}(y_p, y_q) = k(\mathbf{x}_p, \mathbf{x}_q)$$

- noisy observations $y = f(\mathbf{x}) + \epsilon$, $\epsilon \sim N(0, \sigma_n^2)$

$$\text{cov}(y_p, y_q) = k(\mathbf{x}_p, \mathbf{x}_q) + \sigma_n^2 \delta_{pq}$$

$$\text{cov}(\mathbf{y}) = K(X, X) + \sigma_n^2 I$$

- We observe $\mathbf{y}$ and we want to predict $\mathbf{f}_*$:

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

- Predictive distribution

$$\mathbf{f}_* | X, \mathbf{y}, X_* \sim N(\bar{\mathbf{f}}_*, \text{cov}(\bar{\mathbf{f}}_*))$$

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

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

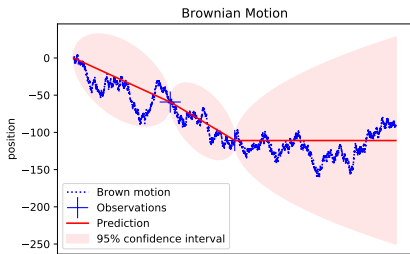
$$\mathbf{f}_* | X, \mathbf{y}, X_* \sim N(\bar{\mathbf{f}}_*, \text{cov}(\bar{\mathbf{f}}_*))$$

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

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

$$\bar{\mathbf{f}}_* \triangleq \mathbb{E} \left[\mathbf{f}_* \mid \begin{bmatrix} 3 \\ 5 \end{bmatrix}, \begin{bmatrix} -5 \\ -11 \end{bmatrix}, [4] \right] = [3, 4] \begin{bmatrix} 3 + \sigma_n^2 & 3 \\ 3 & 5 + \sigma_n^2 \end{bmatrix}^{-1} \begin{bmatrix} -5 \\ -11 \end{bmatrix}$$

$$\text{cov}(\mathbf{f}_*) = \min(4, 4) - [3, 4] \begin{bmatrix} 3 + \sigma_n^2 & 3 \\ 3 & 5 + \sigma_n^2 \end{bmatrix}^{-1} \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$



Predictive distribution

Prediction

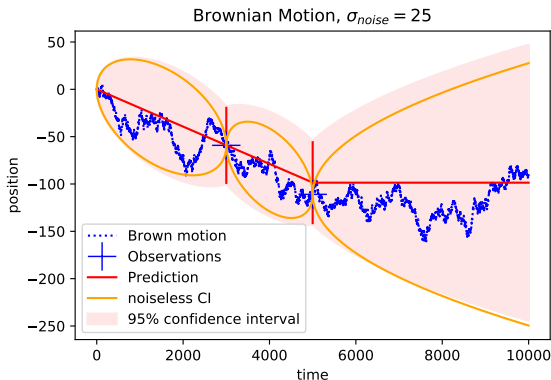
- is a linear function of observations $\mathbf{y}$

- for $\alpha \leftarrow (K + \sigma_n^2 I)^{-1} \mathbf{y}$

- we predict

$$\bar{f}(\mathbf{x}_*) \leftarrow \sum_{i=1}^N \alpha_i k(\mathbf{x}_i, \mathbf{x}_*)$$

- The red vertical bars show the variance due to the observation noise.



Definition (First Set of Kernel Functions!)

- **Radial Basis Function (RBF)** covariance function with the **length scale** parameter ℓ is defined

$$\text{cov}(f(\mathbf{x}_p), f(\mathbf{x}_q)) = \text{RBF}(\mathbf{x}_p, \mathbf{x}_q) = \exp\left(-\frac{1}{2} \frac{|\mathbf{x}_p - \mathbf{x}_q|^2}{\ell^2}\right).$$

- **Constant** covariance function with the **constant** parameter is defined

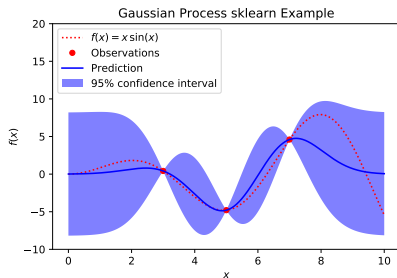
$$\text{cov}(f(\mathbf{x}_p), f(\mathbf{x}_q)) = \text{Constant}(\mathbf{x}_p, \mathbf{x}_q) = \text{constant}.$$

- **Squared exponential (SE)** covariance function with hyperparameters ℓ^2 lengthscale and σ_f^2 signal variance

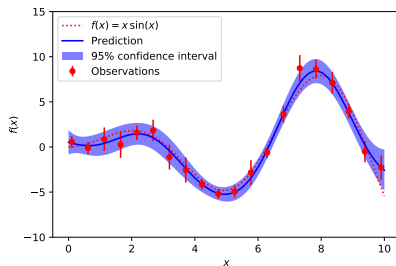
$$\begin{aligned} k(\mathbf{x}_p, \mathbf{x}_q) &= \sigma_f^2 \exp\left(-\frac{1}{2} \frac{|\mathbf{x}_p - \mathbf{x}_q|^2}{\ell^2}\right) \\ &= \text{Constant}(\mathbf{x}_p, \mathbf{x}_q) * \text{RBF}(\mathbf{x}_p, \mathbf{x}_q) \end{aligned}$$

- can be defined as a **product kernel** of the Constant and RBF kernels.
- There is also a **sum kernel** kernel function $+$.

- Noiseless observations.



- The red vertical bars show the variance due to the observation noise.



- The parameters may be fitted by the gradient update.
- The observation noise α may be specific for each observation (right), identically 0 (left) or constant.

```
kernel = C(1.0, (1e-3, 1e3)) * RBF(10, (100e-2, 100e2))
gp = GaussianProcessRegressor(kernel=kernel, alpha=dy ** 2)
gp.fit(X, y)
```

Marginal likelihood

- The parameters may be automatically tuned by gradient maximization of the marginal likelihood.
- Noise-free in sample prediction $\mathbf{f}$ follows: $\mathbf{f} \sim N(\mathbf{0}, \Sigma_{\cdot})$ where $\Sigma_{\cdot} = K(X, X)$.

Lemma

The marginal log likelihood is

- *for noisy-free observations $\mathbf{y} = \mathbf{f}$:*

$$\log p(\mathbf{y}|X) = \log p(\mathbf{f}|X) = -\frac{1}{2}\mathbf{f}^T \Sigma_{\cdot}^{-1} \mathbf{f} - \frac{1}{2} \log |\Sigma_{\cdot}| - \frac{N}{2} \log 2\pi$$

- *For noisy observations $\mathbf{y}|\mathbf{f} \sim N(\mathbf{f}, \sigma_n^2 I)$, $\mathbf{y} \sim N(\mathbf{0}, \Sigma_{\cdot} + \sigma_n^2 I)$*

$$\log p(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^T (\Sigma_{\cdot} + \sigma_n^2 I)^{-1} \mathbf{y} - \frac{1}{2} \log |\Sigma_{\cdot} + \sigma_n^2 I| - \frac{N}{2} \log 2\pi.$$

The noise level may be tuned as well by (sum)adding the **WhiteKernel**.

WhiteKernel = `noise_level` iff we address the same variable ($\mathbf{x}_p, \mathbf{x}_p$), otherwise `WhiteKernel` = 0.

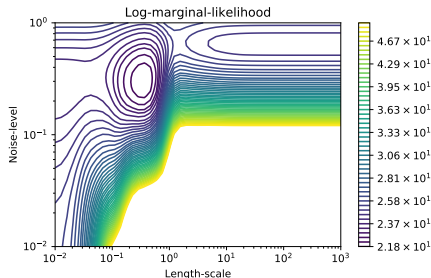
Hyperparameter Fit

Scikitlearn example:

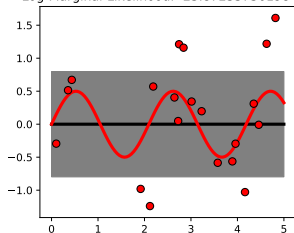
- The log-marginal function has two local maxima.
- The log-marginal maxima corresponds to the two models.

Left initial noise_level=1

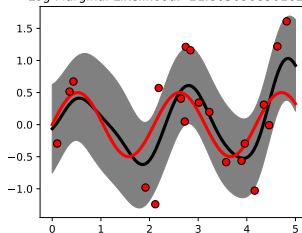
right initial noise_level=0.00001



Initial: $1 \times 10^2 \times \text{RBF}(\text{length_scale}=100) + \text{WhiteKernel}(\text{noise_level}=1)$
num: $0.00316 \times 10^2 \times \text{RBF}(\text{length_scale}=109) + \text{WhiteKernel}(\text{noise_level}=0.00001)$
Log-Marginal-Likelihood: -23.87233736198489



Initial: $1 \times 10^2 \times \text{RBF}(\text{length_scale}=1) + \text{WhiteKernel}(\text{noise_level}=1 \times 10^{-5})$
num: $0.64 \times 10^2 \times \text{RBF}(\text{length_scale}=0.365) + \text{WhiteKernel}(\text{noise_level}=0.00001)$
Log-Marginal-Likelihood: -21.80509089016203



Definition (Further Kernel Functions)

- **ExpSineSquared** kernel function with the parameters **length scale** ℓ and the **periodicity** $p > 0$ (d is the distance) is defined

$$\text{cov}(f(\mathbf{x}_q), f(\mathbf{x}_r)) = \exp \left(-\frac{2 \sin^2(\pi d(\mathbf{x}_q, \mathbf{x}_r)/p)}{\ell^2} \right).$$

Usefull for periodic functions.

- **Dot product** kernel function with the **inhomogeneity** parameter σ_0 is defined

$$\text{cov}(f(\mathbf{x}_p), f(\mathbf{x}_q)) = \sigma_0 + \mathbf{x}_p \cdot \mathbf{x}_q.$$

Useful to capture the trend, often combined with exponential kernel.

- **Rational Quadratic** kernel function with hyperparameters ℓ^2 lengthscale and mixture α

$$k(\mathbf{x}_p, \mathbf{x}_q) = \left(1 + \frac{d(\mathbf{x}_p, \mathbf{x}_q)^2}{2\alpha\ell^2} \right)^{-\alpha}$$

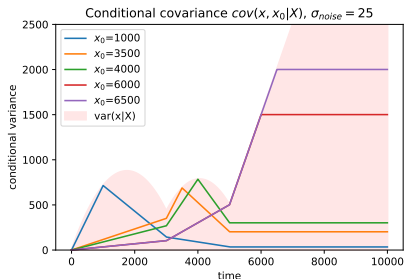
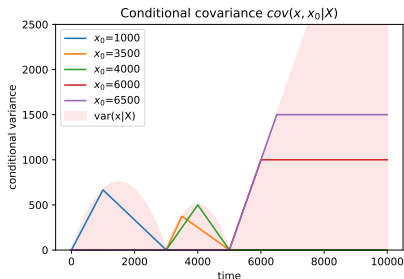
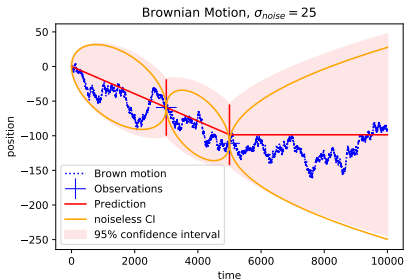
The mixture of many RBF kernel lengthscales.

Conditional Covariance

- Consider the conditional covariance, the relation of two unobserved points x and x_0 .

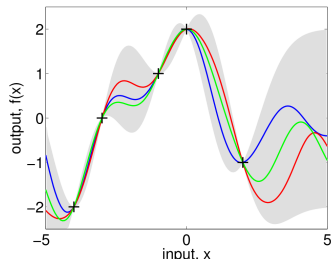
left Noiseless Brown motion example. The covariance is zero outside the x_0 closest observations interval.

right Brown motion with a high noise level.

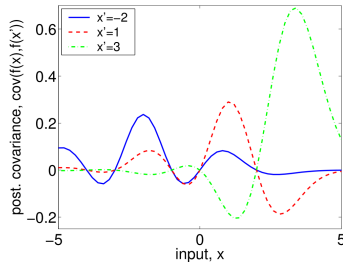


Conditional Covariance Rasmussen Example

- The conditional covariance may be also negative.
- Most kernels have a continuous first derivative. This makes the conditional covariance negative with points on the other side of the closest observation.



(a), posterior



(b), posterior covariance

Matérn

- Most kernel function have many derivatives.
- The Matérn kernel $\nu = 1.5$ ('nu') has only the first derivative. It is able to model less smooth functions.
- As $\nu \rightarrow \infty$, it becomes a RBF kernel.

Definition (Matérn kernel)

The **Matérn** kernel with parameters $\nu = k + \frac{1}{2}$ and ℓ is defined

$$k(\mathbf{x}_p, \mathbf{x}_q) = \frac{1}{2^{\nu-1}\Gamma(\nu)} \left(\frac{\sqrt{2\nu}}{\ell} d(\mathbf{x}_p, \mathbf{x}_q) \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu}}{\ell} d(\mathbf{x}_p, \mathbf{x}_q) \right)$$

The **Modified Bessel functions** (for α not integer, the limit otherwise) are defined

- $I_{\alpha}(x) = \sum_{m=0}^{\infty} \frac{1}{m!\Gamma(m+\alpha+1)} \left(\frac{x}{2}\right)^{2m+\alpha}$
- $K_{\alpha}(x) = \frac{\pi}{2} \frac{I_{-\alpha}(x) - I_{\alpha}(x)}{\sin \alpha \pi}.$

(No need to memorize the formulas.)

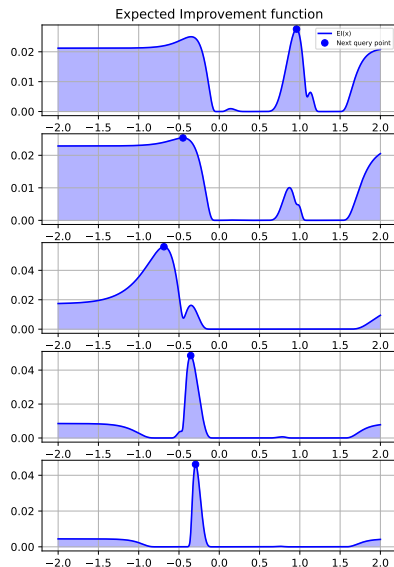
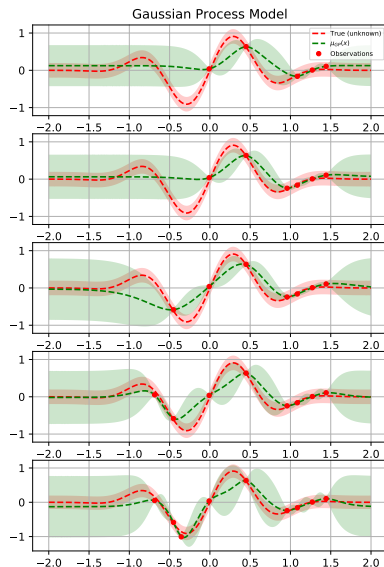
Bayesian Optimization

- Bayesian Optimization is used when
 - We are solving: $x^* = \arg \min_x f(x)$
 - $f(x)$ is a black box function
 - f is expensive to evaluate
 - the evaluations may be noisy.
- If any condition is not true, a better algorithm exists.
- We search the point x to observe.
- scikit-optimize = skopt Python package
- we minimize y and search the maximal probability of improvement
- 'the chance to improve' is expressed by the **Expected improvement** (EI)

Bayesian Optimization Algorithm

- Evaluate y on X , let $y = y(X)$ and calculate conditional means and covariances
- repeat forever
 - $x^{new} = \operatorname{argmax}_x EI(x)$ add x into X
 - Evaluate $y = y(x)$ and add y to y .
 - re-estimate the Gaussian process (the parameters of the covariance).

Bayesian Optimization Example [Skopt]



Expected Improvement Aquisition Function

- we search the point $\mathbf{x}$ to observe
- we minimize y , we already have the training data $X, \mathbf{y}$
- the search the maximal probability of improvement is expressed by the **Expected improvement** (El)

$$\begin{aligned} El(\mathbf{x}) &= \mathbb{E}[(\min(Y(X)) - Y(\mathbf{x}))^+ | Y(X) = \mathbf{y}] \\ &= \mathbb{E}[(\min(\mathbf{y}) - Y(\mathbf{x}))^+ | Y(X) = \mathbf{y}] \end{aligned}$$

this can be solved analytically (Φ cumulative df, ϕ pdf Gaussian distribution):

$$El(\mathbf{x}) = (\min(\mathbf{y}) - \mu(\mathbf{x}))\Phi\left(\frac{\min(\mathbf{y}) - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right) + \sigma(\mathbf{x})\phi\left(\frac{\min(\mathbf{y}) - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right)$$

to maximize $\mathbf{y}$:

$$El(\mathbf{x}) = (\mu(\mathbf{x}) - \max(\mathbf{y}))\Phi\left(\frac{\mu(\mathbf{x}) - \max(\mathbf{y}) - \xi}{\sigma(\mathbf{x})}\right) + \sigma(\mathbf{x})\phi\left(\frac{\mu(\mathbf{x}) - \max(\mathbf{y}) - \xi}{\sigma(\mathbf{x})}\right).$$

- if 'xi' $\xi > 0$ we ignore small improvements.

Parallelization: The Constant Liar

$$\begin{aligned} EI(x) &= \mathbb{E}[(\min(Y(X)) - \min(Y(x^{(n+1)}), Y(x^{(n+2)}), \dots, Y(x^{(n+k)})))^+ | Y(X) = \mathbf{y}] \\ &= \mathbb{E}[(\min(\mathbf{y}) - Y(x))^+ | Y(X) = \mathbf{y}] \end{aligned}$$

it does not have direct formula. It is solved by Markov Chain simulation.

- We estimate the observations y by an estimate (min, max, mean)
- and run the evaluation in parallel.

That means the covariance is correctly estimated, the mean must be corrected later.

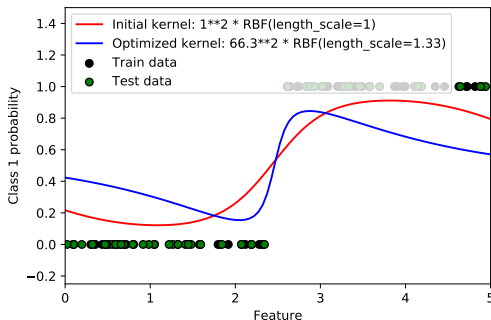
ParBayesianOptimization R package

Definition (Other Acquisition Functions)

- Probability of Improvement: $PI(f(x_*) < \min(\mathbf{y})) = \Phi\left(\frac{\min(\mathbf{y}) - \mu(x_*)}{\sigma(x_*)}\right)$
- Lower Confidence Bound: $LCB(x) = \mu(x) - \kappa \cdot \sigma(x)$.

GP for Classification

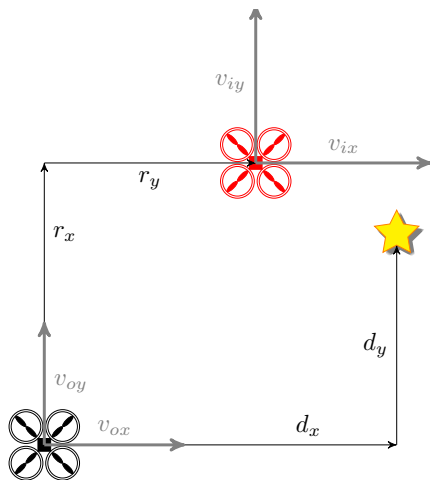
- GP for classification are more complex and 'only an approximation'
- still, it is worth to try the `sklearn.gaussian_process.GaussianProcessClassifier`.
- We estimate a latent function f as before
- we link it to $\langle 0, 1 \rangle$ interval by the sigmoid function (or Φ).
- The log-marginal-likelihood does not have a closed analytical form anymore.
- can be approximated by Hessian matrix, the algorithm works in $O(N^3)$, not too bad.



- Karkus, Hsu, Lee: **QMDP-Net: Deep Learning for Planning under Partial Observability**
<https://proceedings.neurips.cc/paper/2017/file/e9412ee564384b987d086df32d4Paper.pdf>
- Eric Mueller and Mykel J. Kochenderfer : **Multi-Rotor Aircraft Collision Avoidance using Partially Observable Markov Decision Processes**, American Institute of Aeronautics and Astronautics
<https://aviationsystemsdivision.arc.nasa.gov/publications/2016/AIAA-2016-3673.pdf>

POMDP Aircraft Collision Avoidance

- the algorithms designed for fixed-wing aircraft analyze
 - turns
 - vertical maneuvers
- multirotor aircraft (drones) and helicopters can also
 - horizontal plane accelerations
- state 2D, (3D)
 - relative range states $r_x, r_y, (r_z)$
 - velocities for the ownship $v_{ox}, v_{oy}, (v_{oz})$
 - velocities for the intruder $v_{ix}, v_{iy}, (v_{iz})$
 - absolute displacement from the desired trajectory $d_x, d_y, (d_z)$
 - the desired trajectory is normalized to unit velocity in the x axis and zero velocity in the y axis.



MDP Transitions

- the prediction horizon is very short
- updates done every 0.1 to 1 seconds
- simple update equation are sufficient
- not a benefit to using more complex dynamic equations.
- a_x , a_y acceleration by the ownship
- N_* noise to the ownship, intruder, x and y axis
 - $N_o(\mu = 0, 0.30s^{-2})$, $N_i(\mu = 0, 0.45s^{-2})$,
- Bellman update

transition from s with acceleration a to $s^|$

$$Q[s, a] \leftarrow R(s, a) + \gamma \sum_{s^|} T(s^| | s, a) \max_{a^|} Q[s^|, a^|].$$

$$\dot{r}_x = v_{ix} - v_{ox}$$

$$\dot{r}_y = v_{iy} - v_{oy}$$

$$\dot{v}_{ox} = a_x + N_{ox}$$

$$\dot{v}_{oy} = a_y + N_{oy}$$

$$\dot{v}_{ix} = N_{ix}$$

$$\dot{v}_{iy} = N_{iy}$$

$$\dot{d}_x = v_{tx} - v_{ox}$$

$$\dot{d}_y = v_{ty} - v_{oy}$$

- Minimum reward R_{min}
 - collision
 - physically impossible states
 - keeps the sum finite
- we prefer no acceleration
- we prefer long distance to the intruder
- we prefer short distance to the desired trajectory
- K_s, K_T, R_{min} weights was learned, k weights was = 1.

$$R(s, a) = \max \left[R_{min}, -(k_{ax}|a_x| + k_{ay}|a_y|) - K_s \frac{1}{k_{rx}r_x^2 + k_{ry}r_y^2} - K_T(k_{dx}d_x^2 + k_{dy}d_y^2) \right]$$

- offline optimization
 - a few hours for coarse discretization, 1 PC
 - initially stationary intruders
 - intruders moving at uniform velocity with a variety of relative headings angles
 - intruders state and dynamic uncertainty were added to the encounters.
- all values normalized
 - the coarse set contained a total of 765,625 discrete states
 - the finely discretized version contained 9,529,569 states.

State variable	State Description	Discretization
r_x, r_y	Intruder range components	$-15, [-7, -3], -1, 0, 1, [3, 7], 15$
v_{ox}, v_{oy}	ownship velocity components	$-5, -3, -1, 0, 1, 3, 5 \text{ s}^{-1}$
v_{ix}, v_{iy}	intruder velocity components	$-5, [-3], -1, 0, 1, [3], 5 \text{ s}^{-1}$
d_x, d_y	desired trajectory distance	$-10, [-3], -1, 0, 1, [3], 10$

Evaluation Function

- The primary goal is to remain safely separated from the intruder aircraft.
 - $r_{5\%CPA}$ 'the closest point of approach', we allow 5% trajectories a little bit closer.
- Figure: required 1.5 units, never closer than 1.1 units.
- Mean deviation distance from the desired trajectory μ_{dev} .

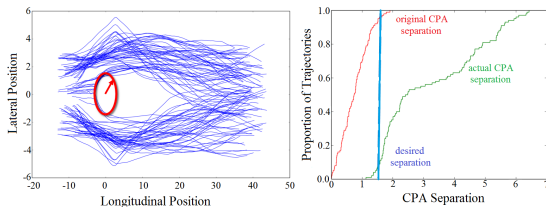
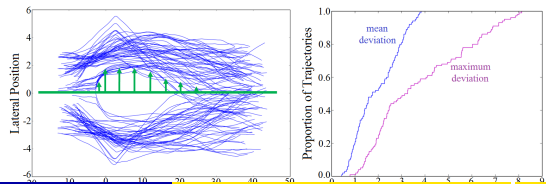


Figure 3: Separation metric used to evaluate the collision avoidance algorithm



Reward Tuning – Bayesian Optimization

- We tune $R_P = (K_T, K_s, R_{min})$
- β weights the two objective functions

$$F(R_P) = (\beta \times (r_{5\%CPA})^{-1} + (1 - \beta) \times \mu_{dev}).$$

- Gaussian process models $F(R_P)$.
- We determine the point at which the objective function is expected to have the largest improvement, $E[I(F(R_P))]$ over that of the current minimum.
- This set of R_P is passed to QMDP to evaluate.
- until convergence.

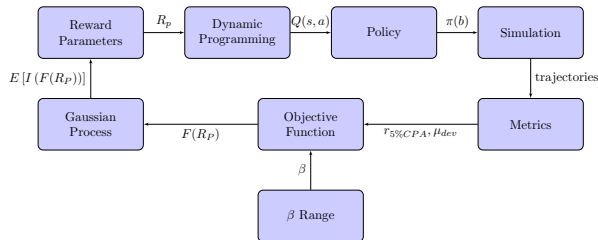
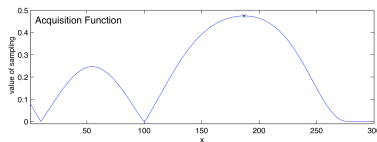
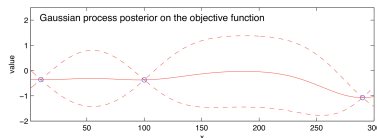


Figure 5: Process for tuning POMDP reward parameters

Bayesian Optimization

- we know QMDP and F values for one or more $\mathbf{x} = R_P$ points
- we search the point $x = R_P^*$ to observe
- we minimize $y = F(R_P^*)$ and search the maximal probability of improvement
- 'the chance to improve' is expressed by the **Expected improvement** (EI)

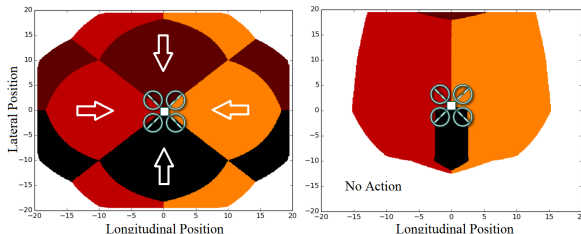


Peter I. Frazier: A Tutorial on Bayesian Optimization, rXiv:1807.02811v1 [stat.ML] 8 Jul 2018

Value Iteration, QMDP Policies

- considerable improvement in convergence speed by initializing by the value of previously evaluated policy
- the value of maximum negative reward influenced the convergence speed
- $\gamma \leftarrow 0.99$ taking hundred iterations to converge.
- smaller γ did not ensure the return to the desired path.

- Figures: Owhship at the origin
- different intruder positions
- policies indicated by color: black=up, red=right
- left: both own and intruder velocities are zero, $d = 0$
- right: owhship is moving in the positive y -axis direction at 1 s^{-1} with zero trajectory error and nominal trajectory matches the velocity.
- The intruder is stationary.



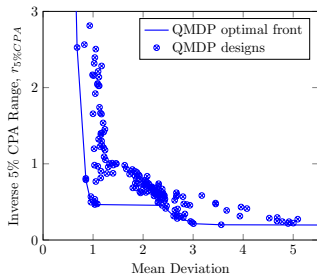
- Uncertainty does not increase with time
- State uncertainty is incorporated only when actions are selected
- a set of potential states is calculated from the observations received at each step.
- the potential states become the beliefs used to select an action.

$$\pi(b) = \max_a \left[\sum_k Q(s^{(k)}, a) b^{(k)} \right]$$

- The value $Q(s^{(k)}, a) b^{(k)}$ approximated from QMDP solutions
 - rectangular interpolation between 2^n nearest neighbor
 - simplex interpolation between $n + 1$ nearest neighbor
 - prior work has found little benefit to using more sophisticated approaches.

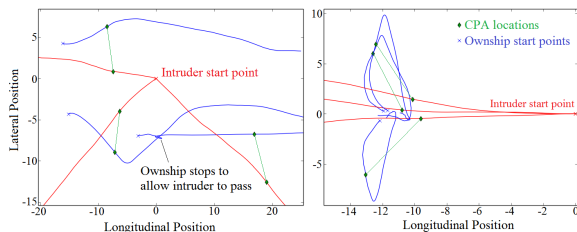
Pareto Optimal frontier

- 194 parameter sets evaluated
- β between 0.01 and 0.99 .
- resulting in nine non-dominated, Pareto-optimal designs.



Human Expert Check

- Left: intruder starts at $(0, 0)$,
 - random heading, fixed velocity of the intruder
 - the ownship starts at the blue cross
- Right: The goal is hovering
 - the intruder comes from the right with the unknown behaviour.



State Discretization

The fine discretization improves the results.

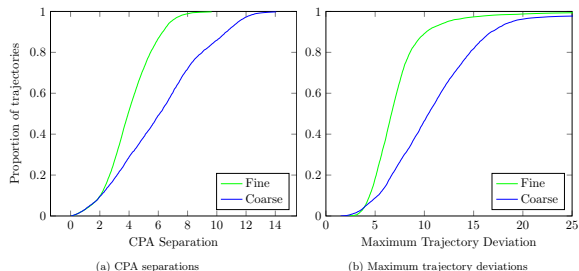


Figure 10: Cumulative distributions of encounter model metrics as a function of state discretization

List of topics

- ① Linear, ridge, lasso regression, k-neares neighbours,(formulas) overfitting, curse of dimensionality, (LARS)
- ② Splines - the base, natural splines, smoothing splines; kernel smoothing: kernel average, Epanechnikov kernel.
- ③ Logistic regression, Linear discriminant analysis, generalized additive models
- ④ Train/test error and data split, square error, 0-1, crossentropy, AIC, BIC,(formulas) crossvalidation, one-leave-out CV, wrong estimate example
- ⑤ decision trees, information gain/entropy/gini, CART pruning,(formulas)
- ⑥ random forest (+bagging), OOB error, Variable importance, boosting (Adaboost(formulas) and gradient boosting), stacking, MARS,
- ⑦ Bayesian learning: MAP, ML hypothesis (formulas), Bayesian optimal prediction, EM algorithm
- ⑧ Clustering: k-means, Silhouette, k-medoids, hierarchical
- ⑨ Apriori algorithm, Association rules, support, confidence, lift
- ⑩ Inductive logic programming basic: hypothesis space search, background knowledge, necessity, sufficiency and consistency of a hypothesis, Aleph
- ⑪ Undirected graphical models, Graphical Lasso procedure, deviance, MRF
- ⑫ Gaussian processes: estimation of the function and its variance (figures, ideas).

List of topics

- ① Linear, ridge, lasso regression, k-neares neighbours,(formulas) overfitting, curse of dimensionality, (LARS)
- ② Splines - the base, natural splines, smoothing splines; kernel smoothing: kernel average, Epanechnikov kernel.
- ③ Logistic regression, Linear discriminant analysis, generalized additive models
- ④ Train/test error and data split, square error, 0-1, crossentropy, AIC, BIC,(formulas) crossvalidation, one-leave-out CV, wrong estimate example
- ⑤ decision trees, information gain/entropy/gini, CART pruning,(formulas)
- ⑥ random forest (+bagging), OOB error, Variable importance, boosting (Adaboost(formulas) and gradient boosting), stacking, MARS,
- ⑦ Bayesian learning: MAP, ML hypothesis (formulas), Bayesian optimal prediction, EM algorithm
- ⑧ Clustering: k-means, Silhouette, k-medoids, hierarchical
- ⑨ Apriori algorithm, Association rules, support, confidence, lift
- ⑩ Inductive logic programming basic: hypothesis space search, background knowledge, necessity, sufficiency and consistency of a hypothesis, Aleph
- ⑪ Undirected graphical models, Graphical Lasso procedure, deviance, MRF
- ⑫ Gaussian processes: estimation of the function and its variance (figures, ideas).

Table of Contents

- 1 Overview of Supervised Learning
- 2 Kernel Methods, Basis Expansion and regularization
- 3 Linear Methods for Classification
- 4 Model Assessment and Selection
- 5 Additive Models, Trees, and Related Methods
- 6 Ensemble Methods
- 7 Bayesian learning, EM algorithm
- 8 Clustering
- 9 Association Rules, Apriori
- 10 Inductive Logic Programming
- 11 Undirected Graphical Models
- 12 Gaussian Processes
- 13 Support Vector Machines
- 14 (PCA Extensions, Independent CA)