Principles for computing viability kernels and resilience values

Laetitia Chapel

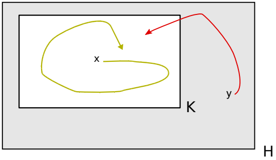
Patres tutorial workshop
22 october 2009

PATRES
PATTERN RESILIENCE
EU FRAMEWORK 6 NEST PROJECT

Introduction

- Control a dynamical system such that
  - Viability perspective: it survives inside a given set of admissible states
  - Resilience perspective: it maintains or restores its property of interest lost after disturbances
- State $x(t)$, controls $u(t)$

$$\begin{cases} x'(t) = \varphi(x(t), u(t)), \text{ for all } t \geq 0 \\ u(t) \in U(x(t)) \subset \mathbb{R}^q \end{cases} \tag{1}$$



Specific framework

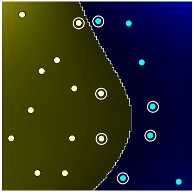
- Viability theory (Aubin)
- Tool: viability algorithm (Saint-Pierre)
- First step: viability kernel approximation
  - But the algorithm suffers the dimensionality curse, in the state and control space
- Second step: compute the resilience values
- Third step: control the system

Idea: sum up the set of points by a function

Support vector machines (SVMs)

- Classification method
  - Inputs: training set, n points with labels $+1/-1$
  - Outputs: linear combination $f(x) = \sum_{i=1}^{n} \alpha_i y_i k(x, x_i) + b$
- Particular solution
  - α_i: influence of a point on the solution $\rightarrow$ parsimonious
  - $k(x, x_i)$ (virtually) maps the points on a feature space
  - non-linear solution in the input space

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



Objectives

- Viability kernel and resilience values approximation algorithms
  - in a high (?) state space
  - in a high control space
- Compact and fast controllers

View points

- Theoretical
- Practical

Sommaire

1. SVMs viability kernel approximation
2. Viability kernels active learning
3. Approximating capture basins with SVMs
4. Resilience computation
5. Conclusion - In practice

Sommaire

1. SVMs viability kernel approximation

2. Viability kernels active learning

3. Approximating capture basins with SVMs

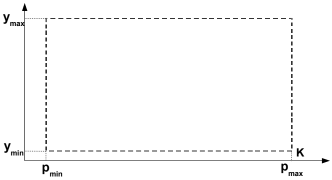
4. Resilience computation

5. Conclusion - In practice

- Simple example of a population growth in a limited space
- Dynamical system

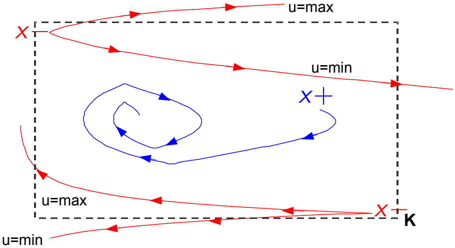
$$\begin{cases} p(t+dt) = p(t) + p(t)y(t)dt \\ y(t+dt) = y(t) + u(t)dt \end{cases} \tag{2}$$

- Under constraints
  - $p \in [p_{min}, p_{max}]$
  - $y \in [y_{min}, y_{max}]$
  - $u \in [-u_{max}, u_{max}]$



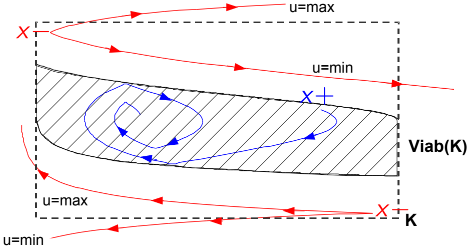
How controlling a dynamical system such that it always remains in K ?

- Approximate the viability kernel of the system



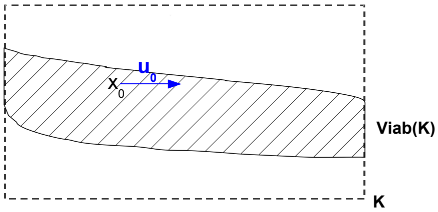
How controlling a dynamical system such that it always remains in K ?

- Approximate the viability kernel of the system



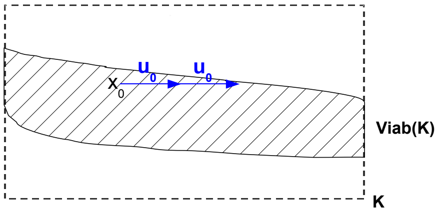
How controlling a dynamical system such that it always remains in K ?

- Viability heavy controller



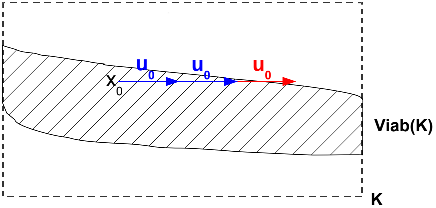
How controlling a dynamical system such that it always remains in K ?

- Viability heavy controller



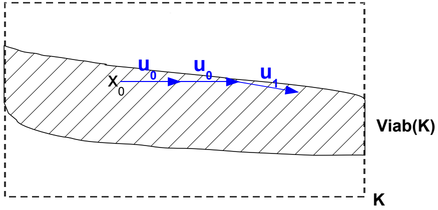
How controlling a dynamical system such that it always remains in K ?

- Viability heavy controller



How controlling a dynamical system such that it always remains in K ?

- Viability heavy controller

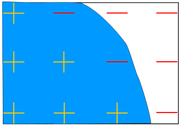


Viability kernel approximation with a classification method

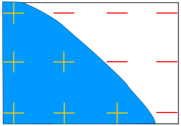
- Iterative algorithm, based on the Saint-Pierre algorithm
- Discretization de K
- Learning set, contains the grid points with labels
  - $+1$ if the point is viable at the next iteration
  - -1 otherwise

Theorem

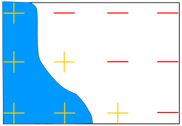
Under some conditions on the learning quality and on the dynamics, the algorithm gives an approximation that converges towards the actual kernel when the grid resolution tends to 0



OK



OK



non OK

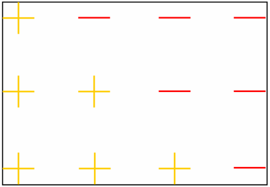
SVMs as a classification procedure: pros

- Allows one to use a optimization method (gradient descent) to find a viable control
- Allows one to work with several time steps at the same time
- The SVMs function can be defined as a controller
- The solution is parcimonious: less points in memory?

- Labels update

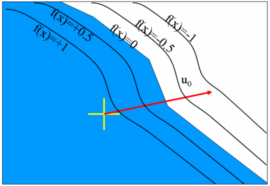
Saint-Pierre

Exhaustive test of the discretized controls



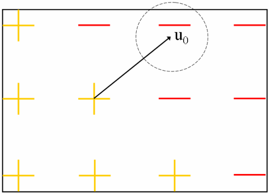
with SVMs

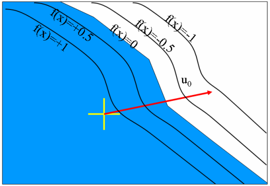
Optimization to find a viable control



- Labels update

Saint-Pierre	with SVMs
Exhaustive test of the discretized controls	Optimization to find a viable control

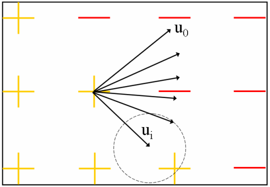




- Labels update

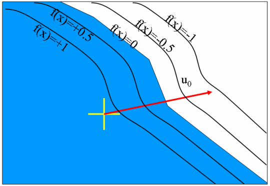
Saint-Pierre

Exhaustive test of the discretized controls



with SVMs

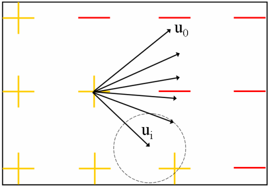
Optimization to find a viable control



- Labels update

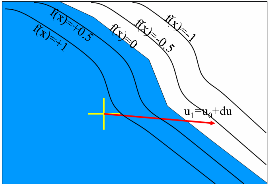
Saint-Pierre

Exhaustive test of the discretized controls



with SVMs

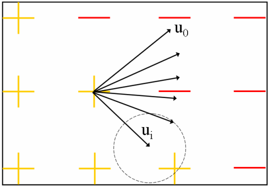
Optimization to find a viable control



- Labels update

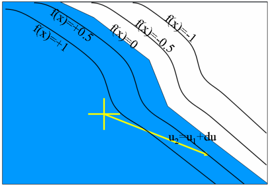
Saint-Pierre

Exhaustive test of the discretized controls



with SVMs

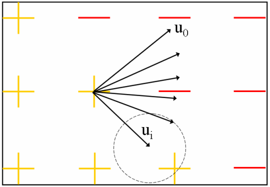
Optimization to find a viable control



- Labels update

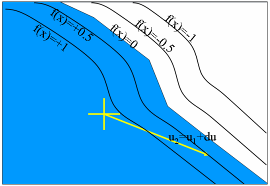
Saint-Pierre

Exhaustive test of the discretized controls



with SVMs

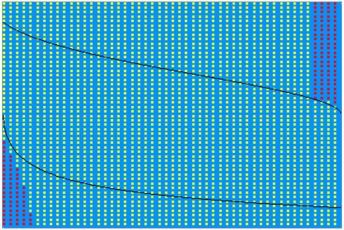
Optimization to find a viable control



Extension to j time steps

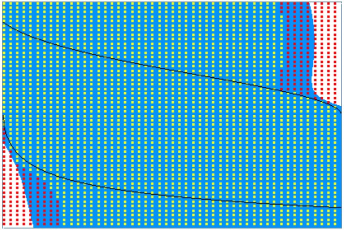
Progressive approximation of the viability kernel

- State space in 2 dimensions, 2601 points, 6 time steps



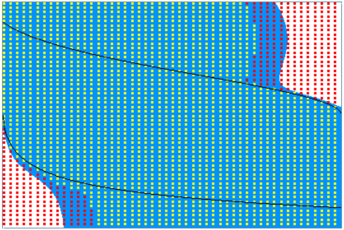
Progressive approximation of the viability kernel

- State space in 2 dimensions, 2601 points, 6 time steps



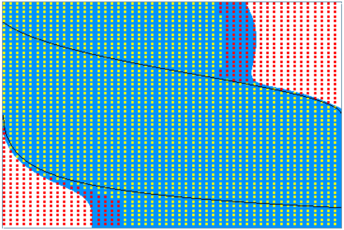
Progressive approximation of the viability kernel

- State space in 2 dimensions, 2601 points, 6 time steps



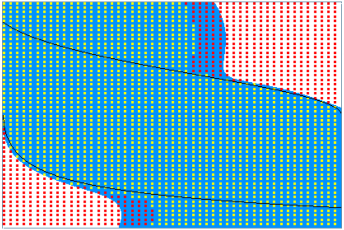
Progressive approximation of the viability kernel

- State space in 2 dimensions, 2601 points, 6 time steps



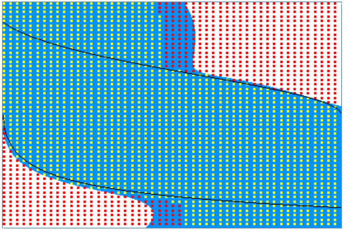
Progressive approximation of the viability kernel

- State space in 2 dimensions, 2601 points, 6 time steps



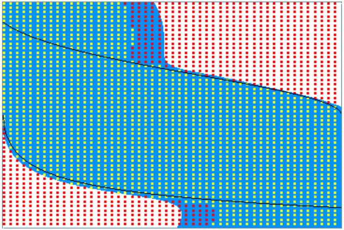
Progressive approximation of the viability kernel

- State space in 2 dimensions, 2601 points, 6 time steps



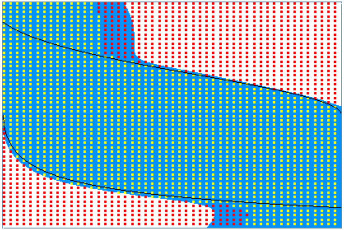
Progressive approximation of the viability kernel

- State space in 2 dimensions, 2601 points, 6 time steps



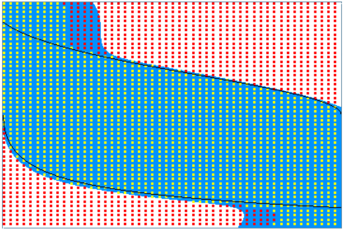
Progressive approximation of the viability kernel

- State space in 2 dimensions, 2601 points, 6 time steps



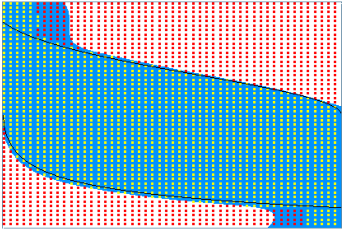
Progressive approximation of the viability kernel

- State space in 2 dimensions, 2601 points, 6 time steps



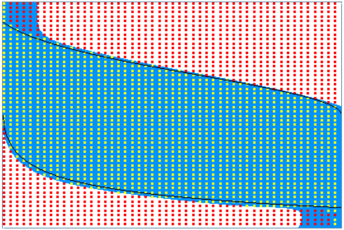
Progressive approximation of the viability kernel

- State space in 2 dimensions, 2601 points, 6 time steps



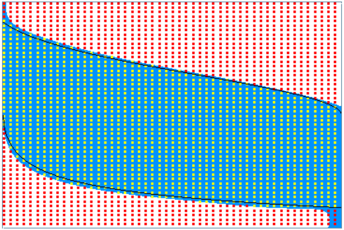
Progressive approximation of the viability kernel

- State space in 2 dimensions, 2601 points, 6 time steps



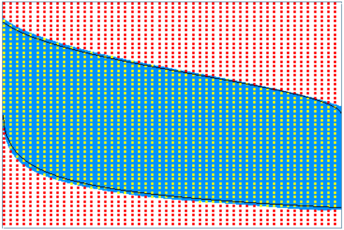
Progressive approximation of the viability kernel

- State space in 2 dimensions, 2601 points, 6 time steps



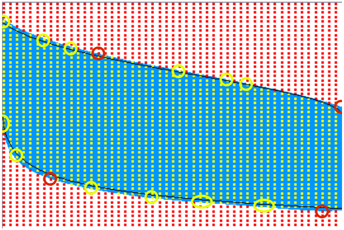
Progressive approximation of the viability kernel

- State space in 2 dimensions, 2601 points, 6 time steps



Progressive approximation of the viability kernel

- State space in 2 dimensions, 2601 points, 6 time steps

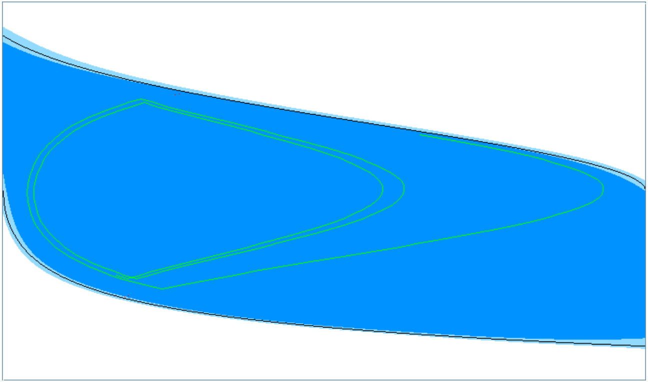


- 12 iterations, 19 SV

SVMs viability heavy controller

- Same control u_0 until the next state reaches $f(x) < \Delta$
- Find a viable control, using a gradient descent on f
- More or less cautious controller, anticipating on several time steps

Controller example (5 steps anticipation)



Conclusion [Viability kernel approximation]

- Use of SVMs to approximate a viability kernel
  - defines compact controllers
  - allows one to deal with problems on high control space (thanks to the control optimization)
  - doesn't allow to deal with problems on high state space (discretization of the state space)

Sommaire

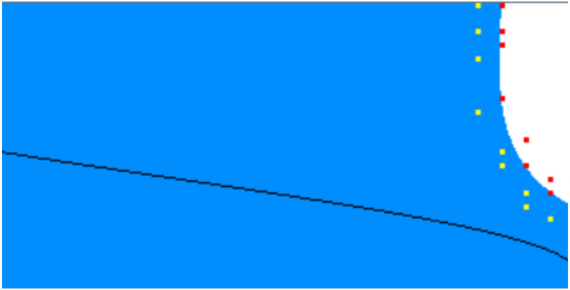
1. SVMs viability kernel approximation
2. Viability kernels active learning
3. Approximating capture basins with SVMs
4. Resilience computation
5. Conclusion - In practice

- active learning: limits the number of points to label / the training set size
  - label points is costly
  - the grid size is exponential with the dimension
  - learning a SVM function is quadratic with the training set size
- We use the parsimonious property of the SVMs

- Aim: use a number of points close to the number of SVs
- We progressively add the points the more likely to be SVs (in couple)
- Question: which points to chose? $\longrightarrow$ we focus on the boundary
- We use a virtual multi-resolution grid

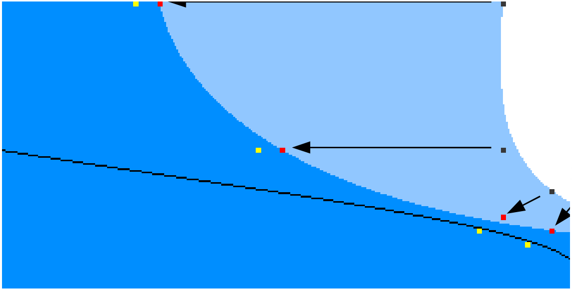
In action

- Multi-resolution grid of depth 3



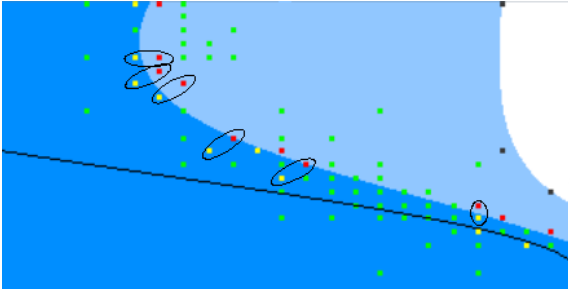
In action

- Multi-resolution grid of depth 3



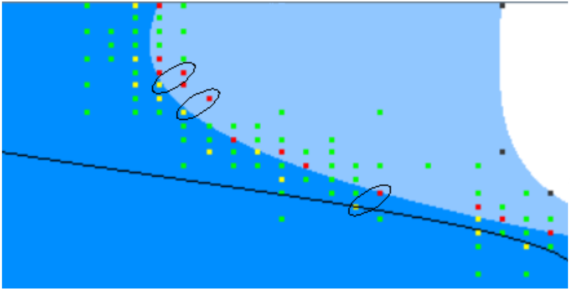
In action

- Multi-resolution grid of depth 3



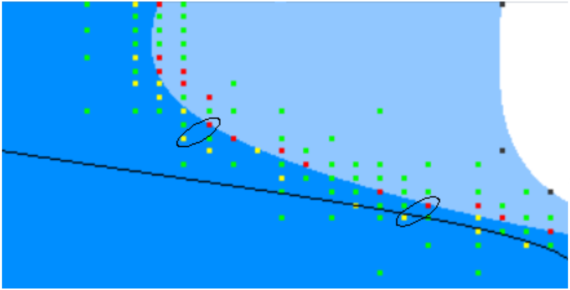
In action

- Multi-resolution grid of depth 3



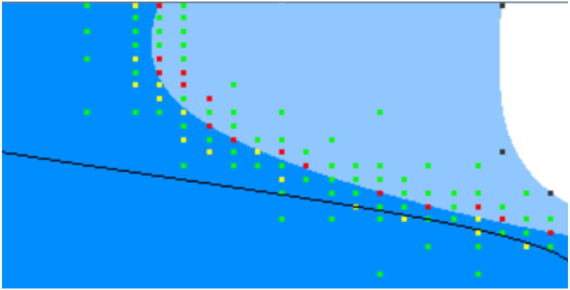
In action

- Multi-resolution grid of depth 3



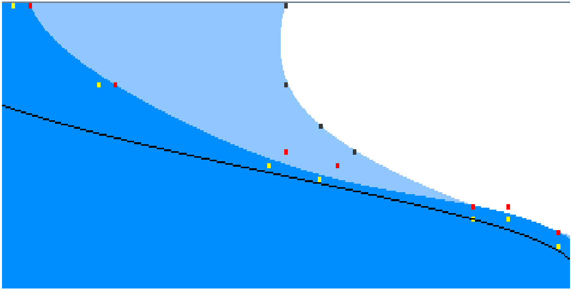
In action

- Multi-resolution grid of depth 3

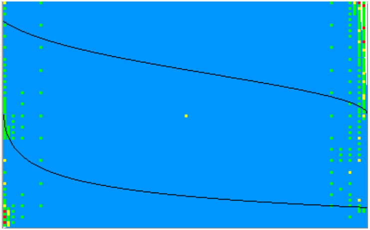


In action

- Multi-resolution grid of depth 3

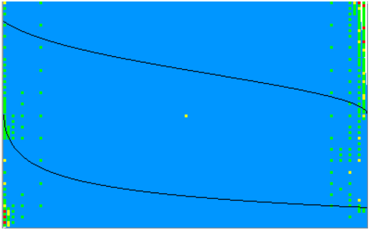


- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



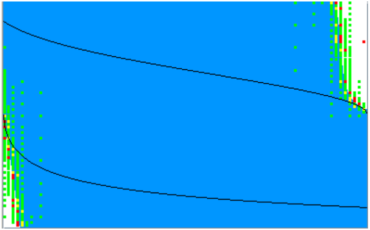
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



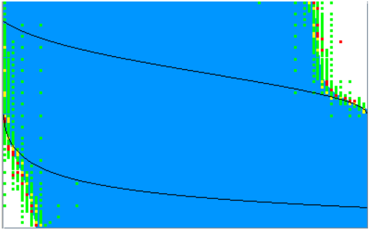
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



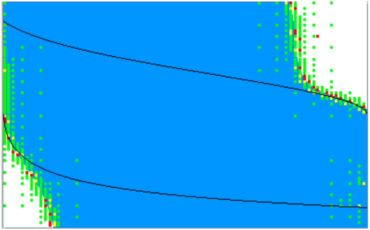
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



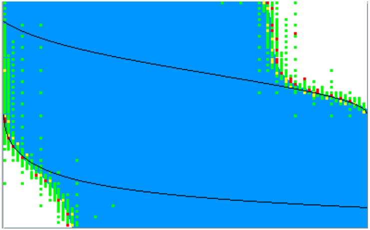
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



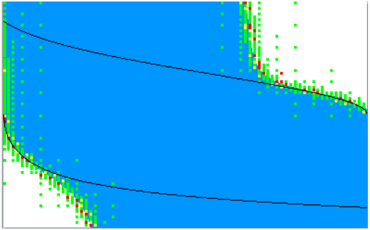
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



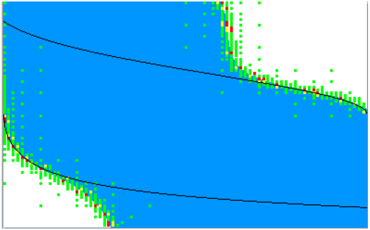
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



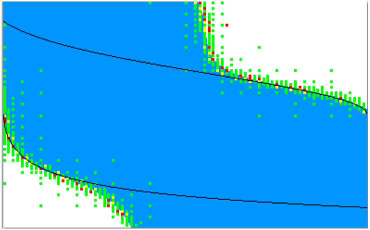
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



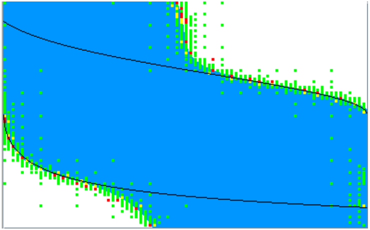
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



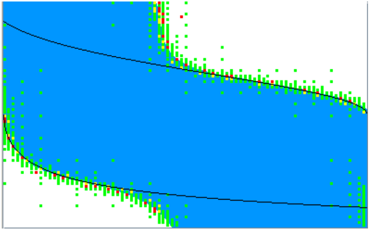
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 → 6561 points on the whole grid



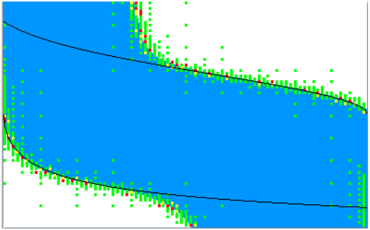
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



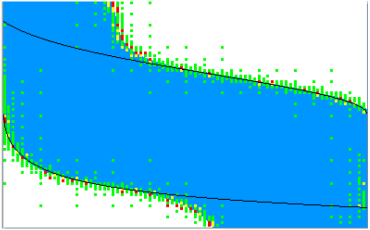
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



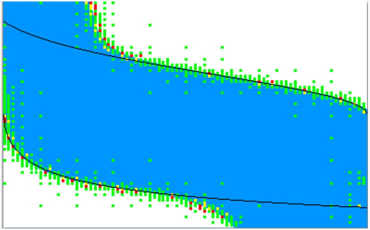
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



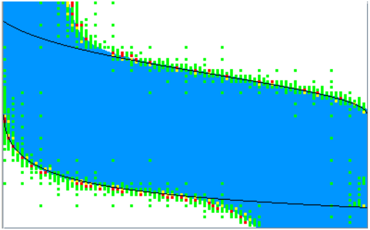
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



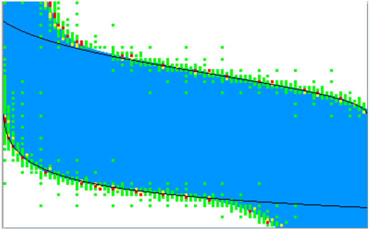
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

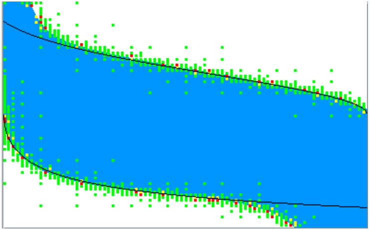
- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

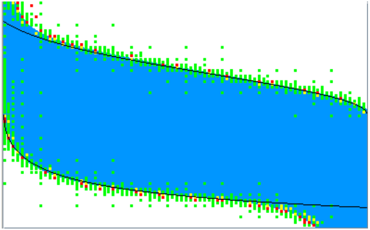
Viability kernels active learning
Application example (population)

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



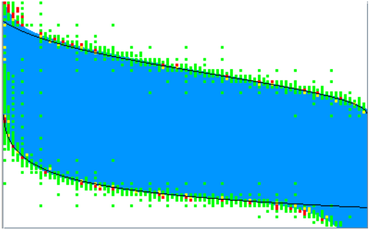
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



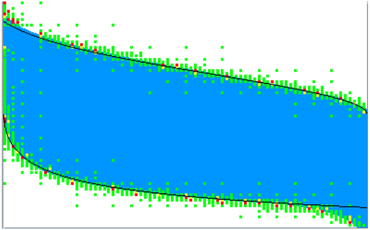
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



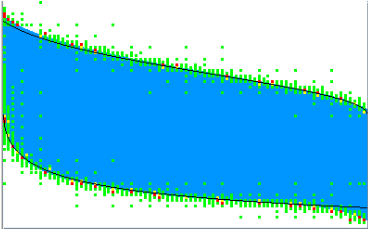
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



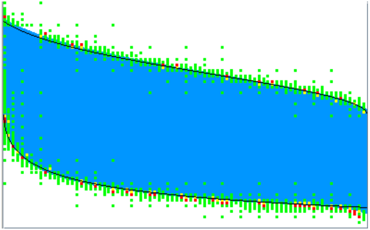
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



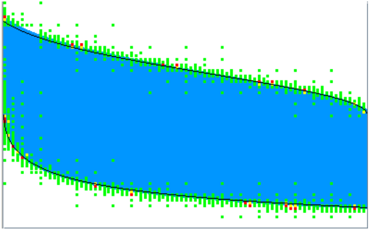
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 → 6561 points on the whole grid



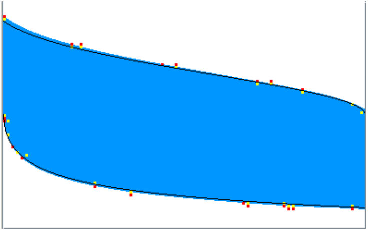
- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

- 11 points by dimension, grid of depth 4 $\rightarrow$ 6561 points on the whole grid



- 6 SVMs learning by iteration
- 28 SV, 761 (12%) max labeled, 124 (2%) max in $\mathcal{S}$

Conclusion [Active learning]

- Allows one to limit the size of the training set size
  - but the approximation time is still exponential
  - produces a fast and compact controller

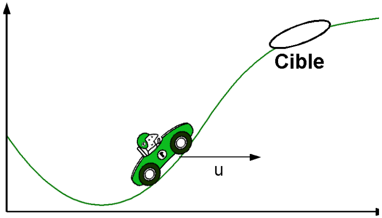
Sommaire

1. SVMs viability kernel approximation
2. Viability kernels active learning
3. Approximating capture basins with SVMs
4. Resilience computation
5. Conclusion - In practice

- The car has to reach as fast as possible the top of the hill, while staying in a given state space
- Dynamical system

$$\begin{cases} p(t+dt) = p(t) + v(t)dt \\ v(t+dt) = v(t) + f(u(t))dt \end{cases} \qquad (3)$$

- Under constraints
  - $p \in [p_{min}, p_{max}]$
  - $v \in [v_{min}, v_{max}]$
  - $u \in [-u_{max}, u_{max}]$



- Capture basin at time t_{Max}: set of initial states that can reach the target before time $\tau \leq t_{Max}$, while staying in K
- Algorithm: addition of one dimension, the time $\tau' = -1$
- Approximation of the viability kernel of the extended system, with $K \times [0, t_{Max}]$

$$-\text{if } x \notin C \quad F_C(x,\tau) = \begin{cases} x'(t) = F(x(t), u(t)) \\ \tau'(t) = -1 \end{cases} \tag{4}$$

$$-\text{if } x \in C \quad F_C(x,\tau) = 0$$

Approximation algorithm in the initial state space

- Iterative algorithm, based on the viability kernel algorithm
- Initialization: $+1$ if $x \in C$, -1 otherwise
- Learning set: points of the grid with label
  - $+1$ if there exists a control that will lead the point to the current target at the next time step
  - -1 otherwise

Theorem [outer approximation]

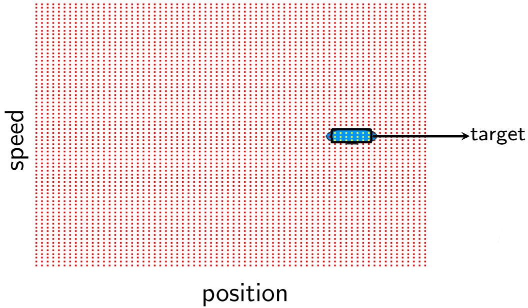
Under some conditions on the learning quality and on the dynamics, the algorithm provides a capture basin approximation that converge towards the actual capture basin when the resolution of the grid tends to 0

Theorem [inner approximation]

Under some conditions on the learning quality and on the dynamics, the algorithm provides a capture basin approximation that converge towards the actual capture basin when the resolution of the grid tends to 0

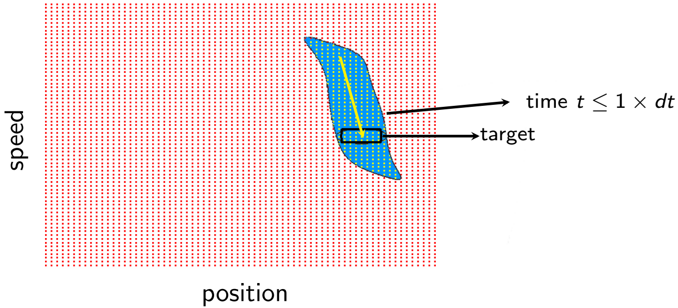
Progressive outer approximation of the capture basin

- 2 dimension state space, grid of 5041 points, 8 time steps



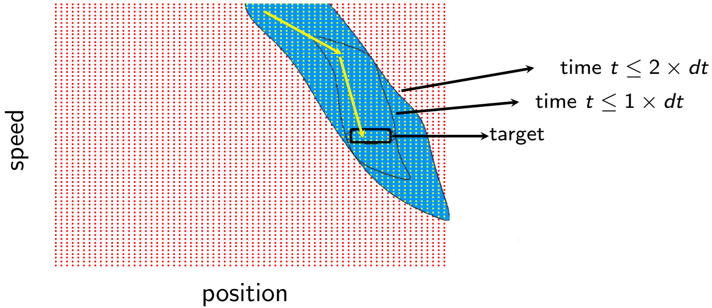
Progressive outer approximation of the capture basin

- 2 dimension state space, grid of 5041 points, 8 time steps



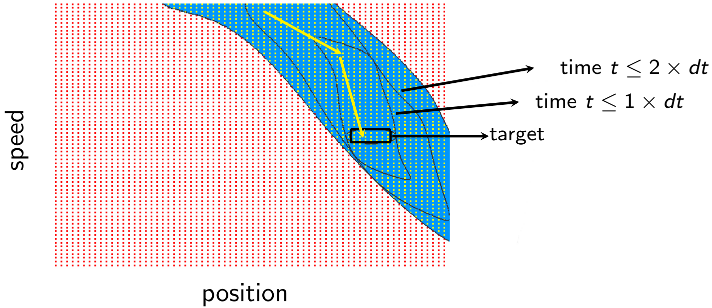
Progressive outer approximation of the capture basin

- 2 dimension state space, grid of 5041 points, 8 time steps



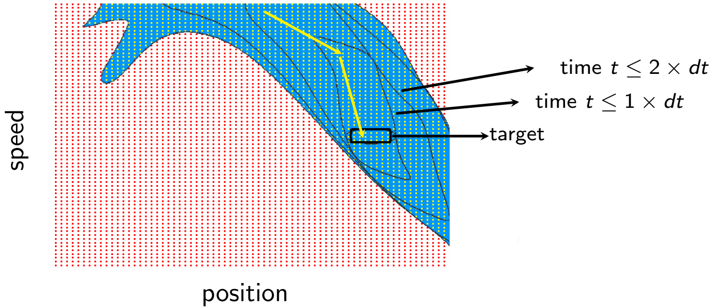
Progressive outer approximation of the capture basin

- 2 dimension state space, grid of 5041 points, 8 time steps



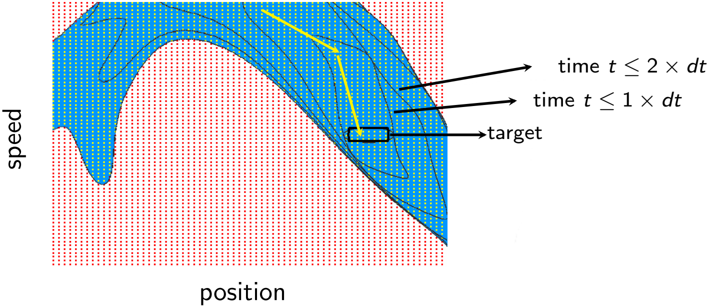
Progressive outer approximation of the capture basin

- 2 dimension state space, grid of 5041 points, 8 time steps



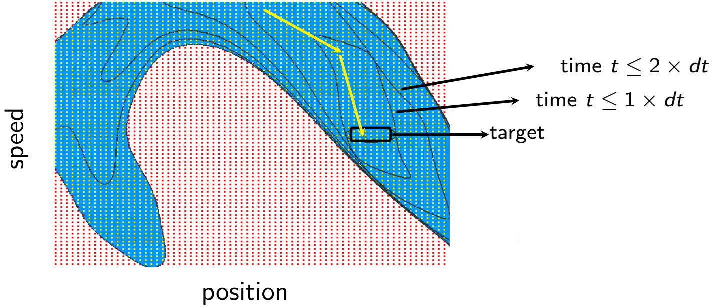
Progressive outer approximation of the capture basin

- 2 dimension state space, grid of 5041 points, 8 time steps



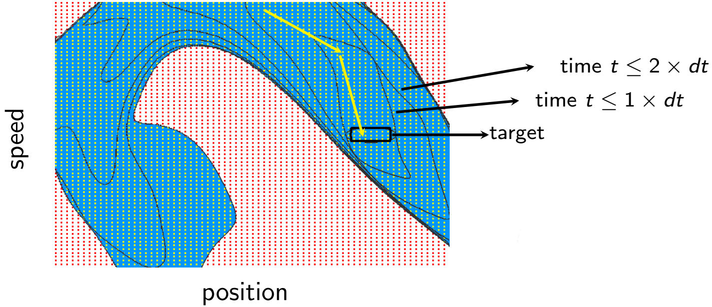
Progressive outer approximation of the capture basin

- 2 dimension state space, grid of 5041 points, 8 time steps



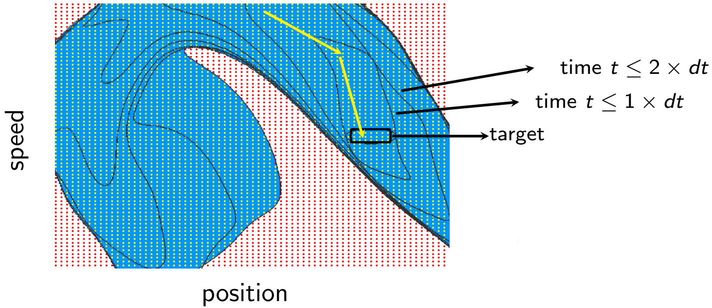
Progressive outer approximation of the capture basin

- 2 dimension state space, grid of 5041 points, 8 time steps



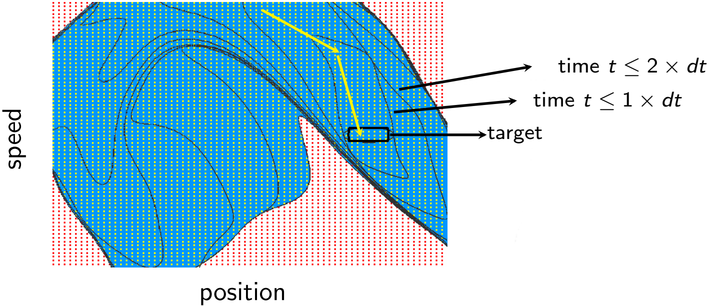
Progressive outer approximation of the capture basin

- 2 dimension state space, grid of 5041 points, 8 time steps



Progressive outer approximation of the capture basin

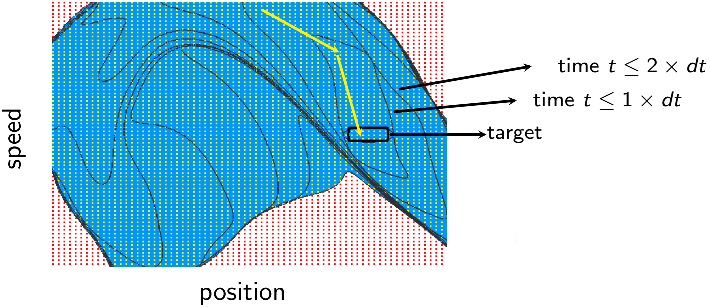
- 2 dimension state space, grid of 5041 points, 8 time steps



Approximating capture basins with SVMs
Application example

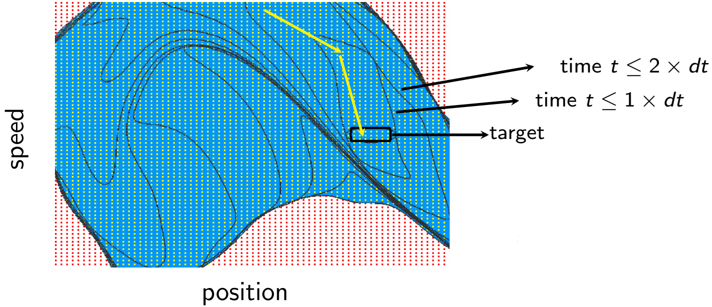
Progressive outer approximation of the capture basin

- 2 dimension state space, grid of 5041 points, 8 time steps



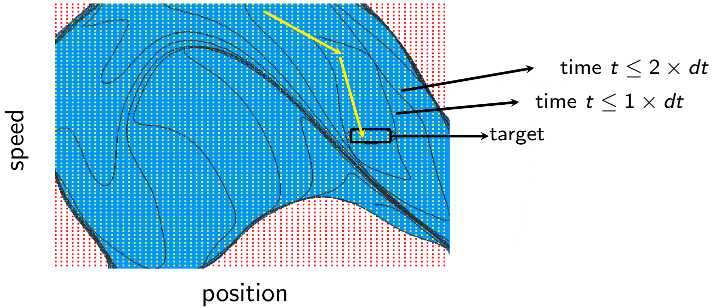
Progressive outer approximation of the capture basin

- 2 dimension state space, grid of 5041 points, 8 time steps



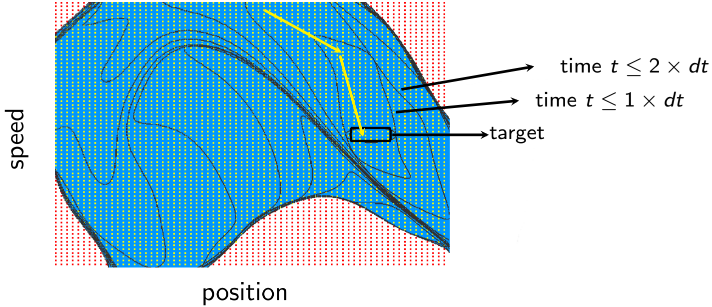
Progressive outer approximation of the capture basin

- 2 dimension state space, grid of 5041 points, 8 time steps



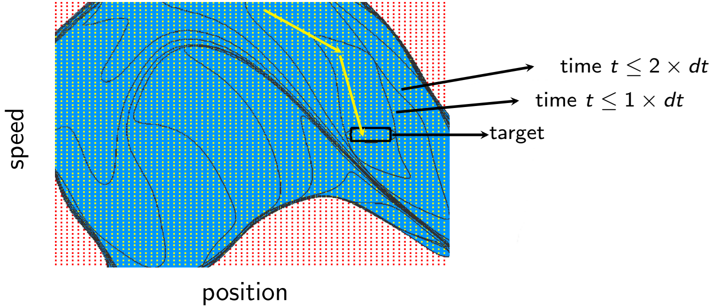
Progressive outer approximation of the capture basin

- 2 dimension state space, grid of 5041 points, 8 time steps



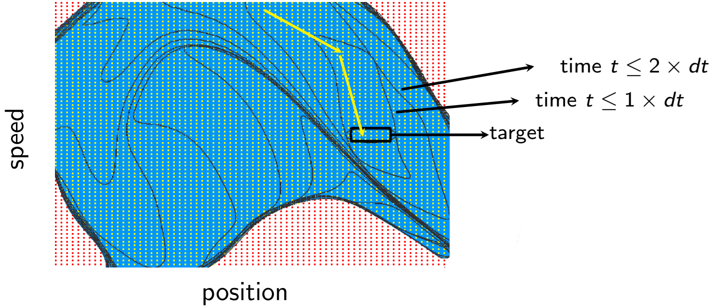
Progressive outer approximation of the capture basin

- 2 dimension state space, grid of 5041 points, 8 time steps



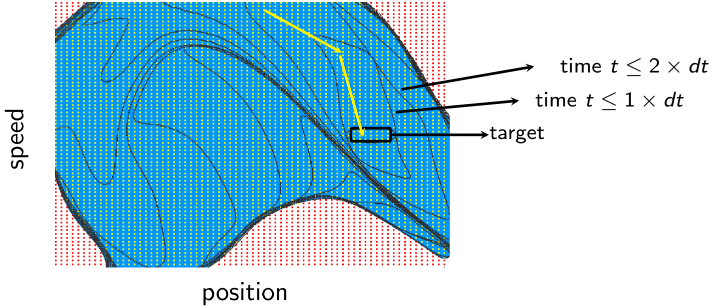
Progressive outer approximation of the capture basin

- 2 dimension state space, grid of 5041 points, 8 time steps



Progressive outer approximation of the capture basin

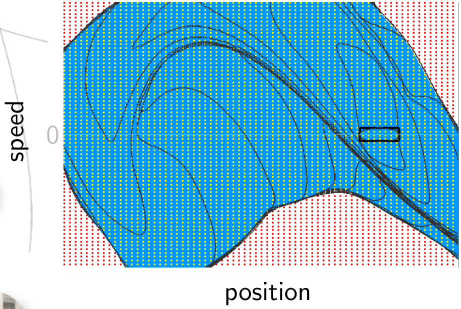
- 2 dimension state space, grid of 5041 points, 8 time steps

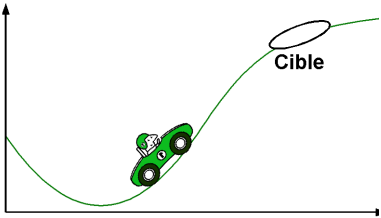


- Capture basin active learning

Contrôleur optimal

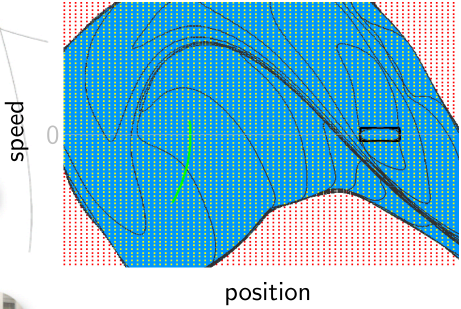
- Optimal control from the inner approximation

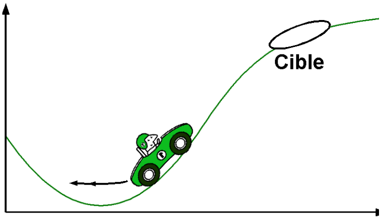




Contrôleur optimal

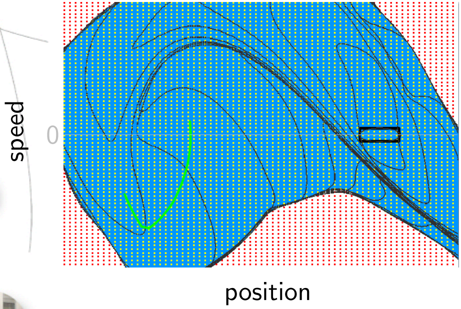
- Optimal control from the inner approximation

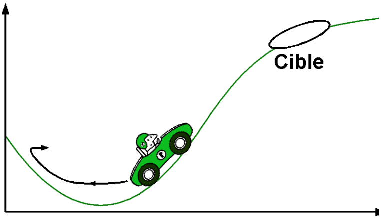




Contrôleur optimal

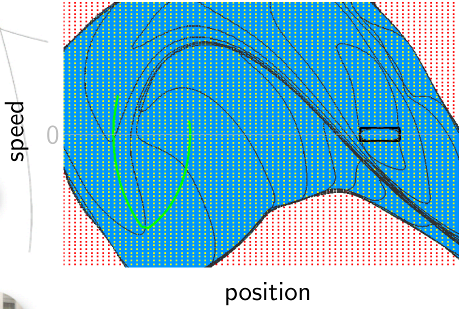
- Optimal control from the inner approximation

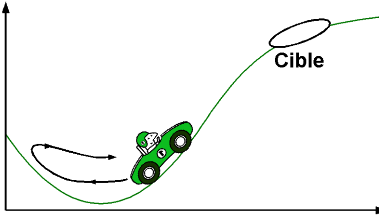




Contrôleur optimal

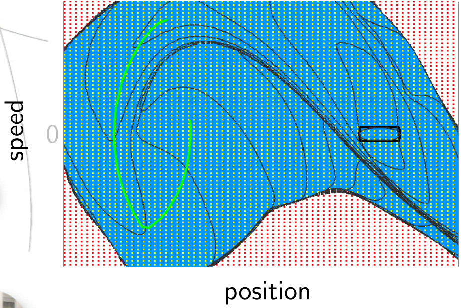
- Optimal control from the inner approximation

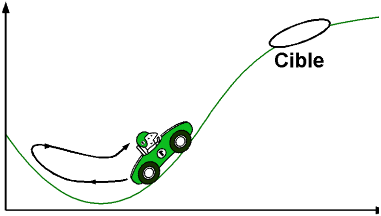




Contrôleur optimal

- Optimal control from the inner approximation

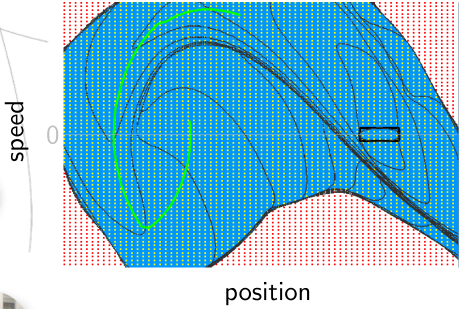


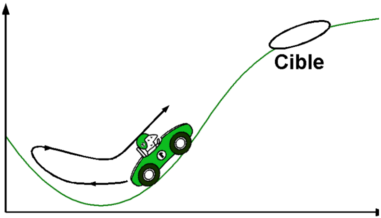


Approximating capture basins with SVMs
Contrôleur

Contrôleur optimal

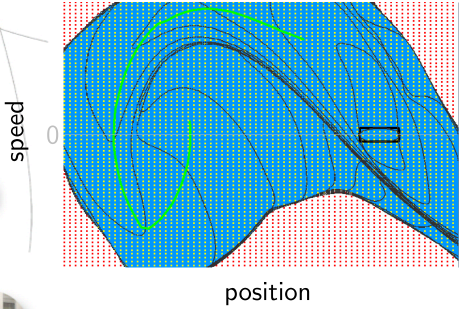
- Optimal control from the inner approximation

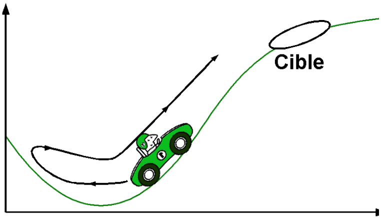




Contrôleur optimal

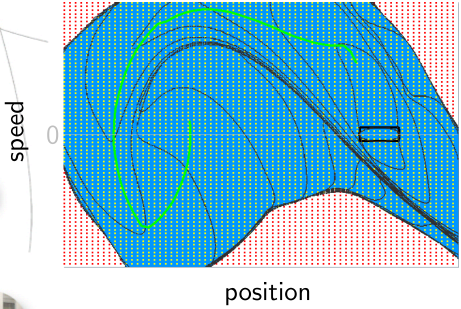
- Optimal control from the inner approximation

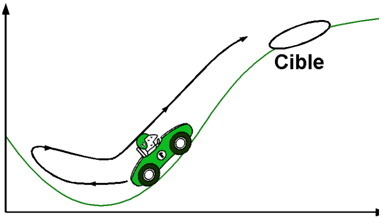




Contrôleur optimal

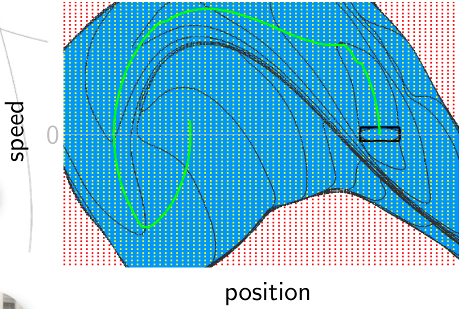
- Optimal control from the inner approximation

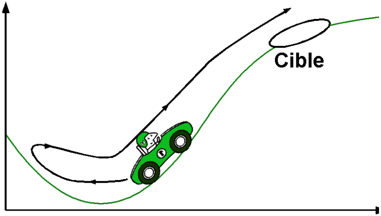




Contrôleur optimal

- Optimal control from the inner approximation





Conclusion [Capture basin approximation]

- Two versions of the algorithm
  - allows one to define a controller that guarantee to reach the target
  - allows one to work in high dimensional control space
  - doesn't allows one to deal with high dimensional state space

Sommaire

1. SVMs viability kernel approximation
2. Viability kernels active learning
3. Approximating capture basins with SVMs
4. Resilience computation
5. Conclusion - In practice

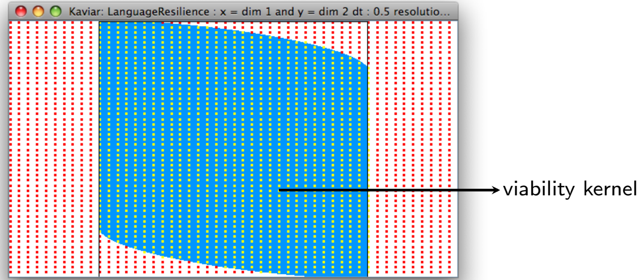
Resilience computation
Extension of the capture basin algorithm

Extension to resilience values computation

- Definition [Martin] : inverse of the restoration cost of the properties of interest lost after disturbances
- Property of interest: viability constraint set
- Under some conditions, we can use the capture basin approximation algorithm
  - target: viability kernel
  - set of states that can go back to the target with a cost $c \leq c_{max}$

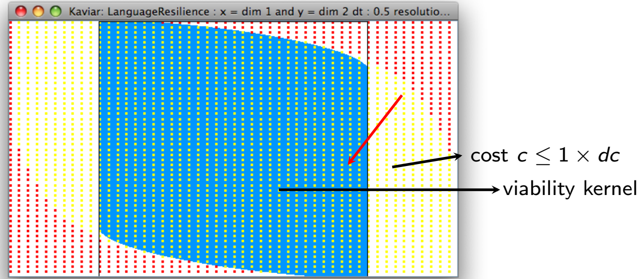
Progressive inner approximation of the resilience values

- AS model



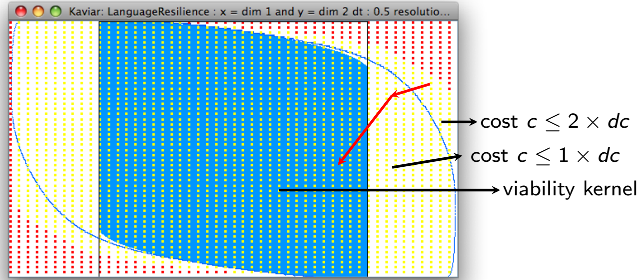
Progressive inner approximation of the resilience values

- AS model



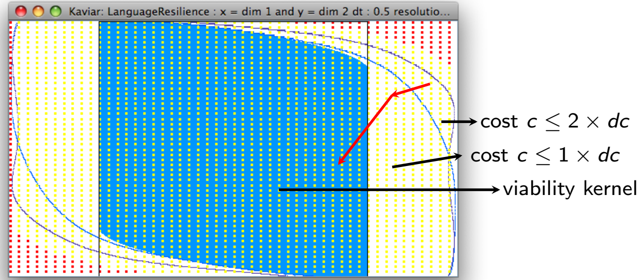
Progressive inner approximation of the resilience values

- AS model



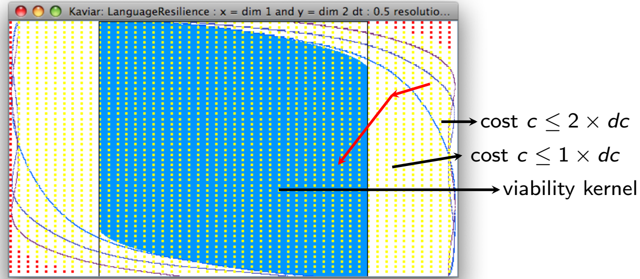
Progressive inner approximation of the resilience values

- AS model



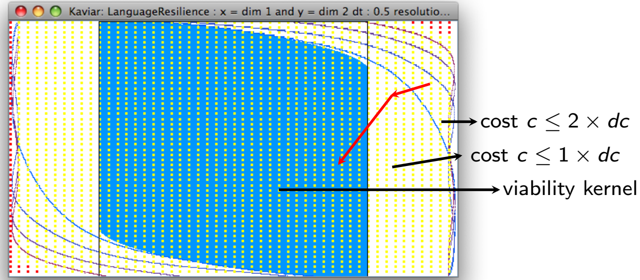
Progressive inner approximation of the resilience values

- AS model



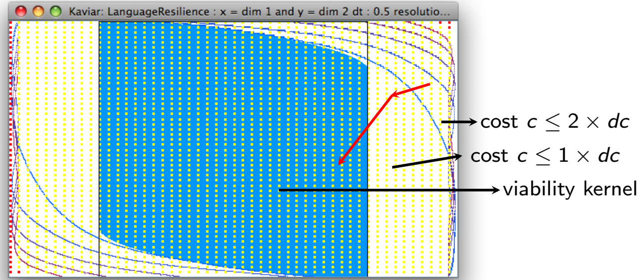
Progressive inner approximation of the resilience values

- AS model



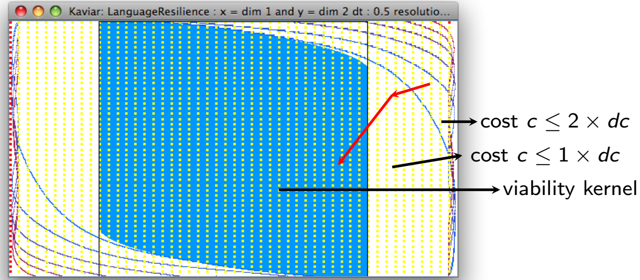
Progressive inner approximation of the resilience values

- AS model



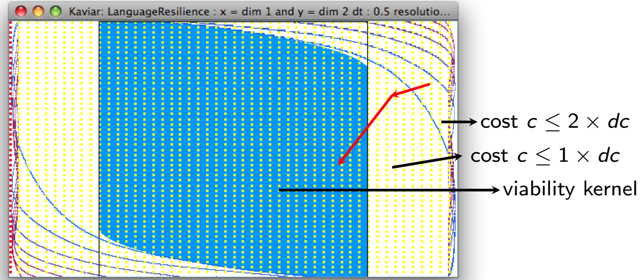
Progressive inner approximation of the resilience values

- AS model



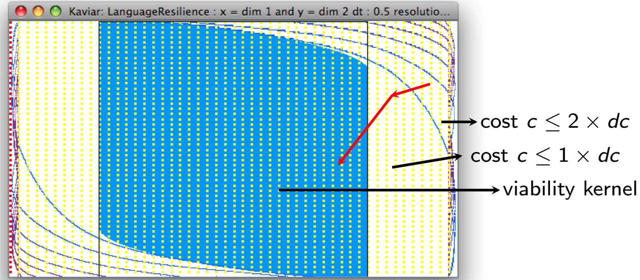
Progressive inner approximation of the resilience values

- AS model



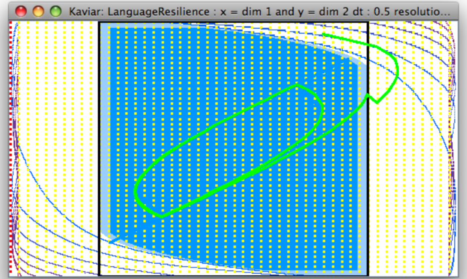
Progressive inner approximation of the resilience values

- AS model



Progressive inner approximation of the resilience values

- AS model



Sommaire

1. SVMs viability kernel approximation
2. Viability kernels active learning
3. Approximating capture basins with SVMs
4. Resilience computation
5. Conclusion - In practice

Conclusion - In practice

How to obtain a good approximation?

- Know your system and dynamics! especially the speed of the dynamics, in order to define good time step, grid size, dt etc.
- Especially with the resilience computation

How to know if I can rely on my approximation?

- Use the controller! Starting from several starting points, you can produce a trajectory that remains inside the kernel + leaves K

Principles for computing viability kernels and resilience values

Laetitia Chapel

Patres tutorial workshop
22 october 2009

PATRES
PATTERN RESILIENCE
EU FRAMEWORK 6 NEST PROJECT