SVM viability controller active learning

Laetitia Chapel   Guillaume Deffuant

Laboratory of Engineering for Complex Systems (LISC)

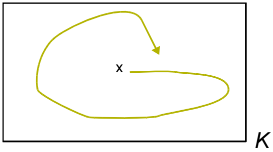
workshop krl - ICML 2006, June 29, 2006

Cemagref

Introduction

- We want to control a dynamical system such that it can survive inside a given set of admissible states
- State $x(t)$, controls $u(t)$, in discrete time

$$\begin{cases} x(t+dt) = x(t) + \varphi(x(t), u(t))dt, & \text{for all t} \geq 0 \\ u(t) \in U(x(t)) \end{cases} \tag{1}$$



- Reinforcement learning problem, negative reward outside K

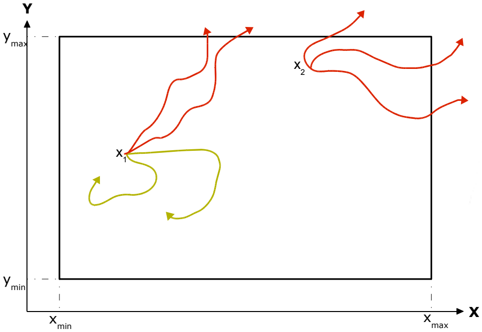
Outline

1. Viability theory
2. SVM viability controller
3. SVM viability controller active learning
4. Discussion and perspectives

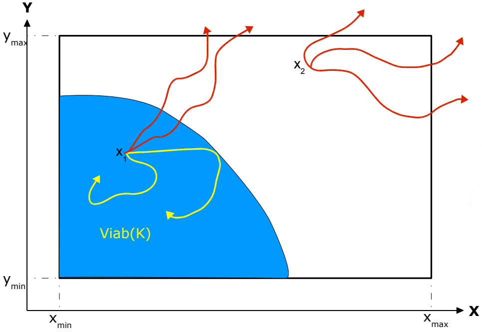
Outline

1. Viability theory
2. SVM viability controller
3. SVM viability controller active learning
4. Discussion and perspectives

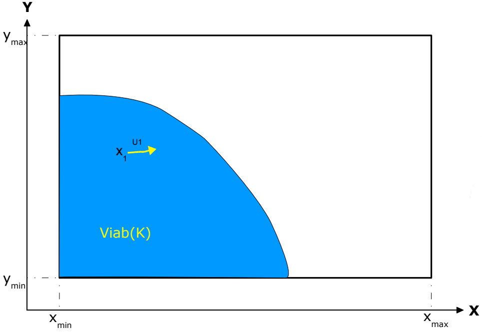
- Viable state: There exists at least one control function for which the whole trajectory remains in K



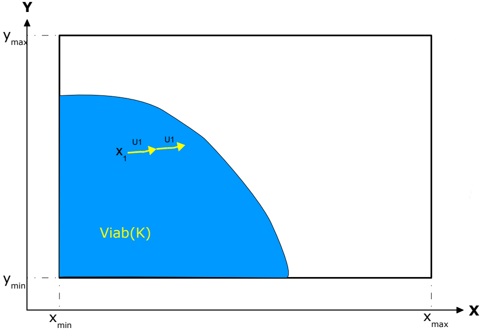
- Viability kernel: Set of all viable states



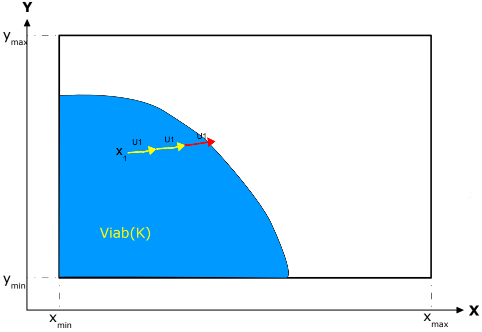
- Viability controller: The viability kernel is instrumental to define viable control policies



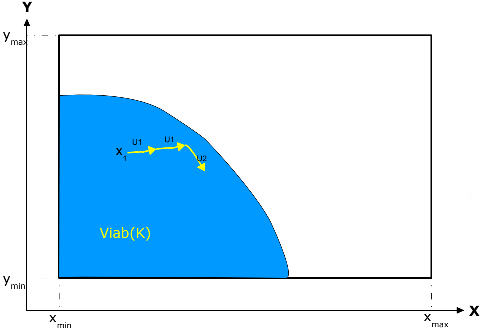
- Viability controller: The viability kernel is instrumental to define viable control policies



- Viability controller: The viability kernel is instrumental to define viable control policies



- Viability controller: The viability kernel is instrumental to define viable control policies



- There is no explicit formula to determine the viability kernel
- Saint-Pierre: based on the discretization of K. But:
  - not convenient to manipulate
  - control space dimensionality curse
  - state space dimensionality curse
- Ultra-Bee: using a value function. But:
  - only for state space of 2 dimensions

- There is no explicit formula to determine the viability kernel
- Saint-Pierre: based on the discretization of K. But:
  - not convenient to manipulate
  - control space dimensionality curse

    } SVM
  - state space dimensionality curse $\rightarrow$ Active learning
- Ultra-Bee: using a value function. But:
  - only for state space of 2 dimensions

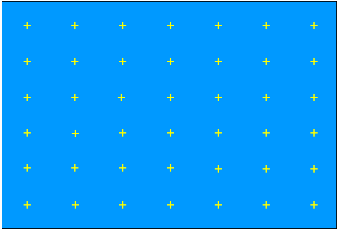
Outline

1. Viability theory
2. SVM viability controller
3. SVM viability controller active learning
4. Discussion and perspectives

- Algorithm based on the discretization of K
- Iterative approximation of Viab(K)
- Points of the grid viable at the next step $\longrightarrow$ label +1
  the others $\longrightarrow$ label -1
- SVM function provides a kind of barrier function on the viability kernel boundary, which enables to use gradient techniques to find a viable control

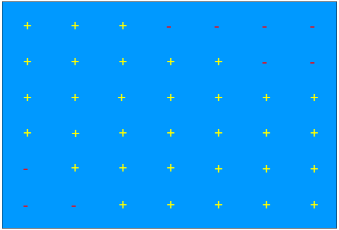
Initialization

- Discretization of the state space



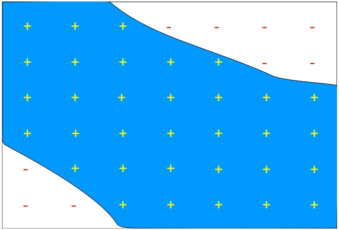
Initialization

- Initialization of non-viable examples



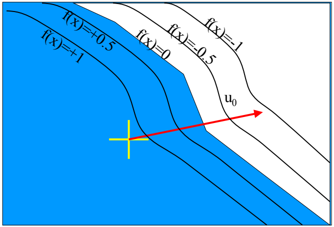
Iteration $n+1$

- SVM$_n$ is available



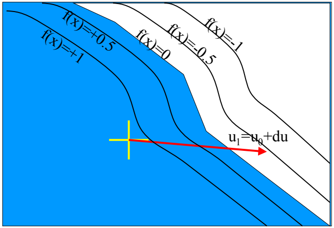
Iteration $n + 1$

- Gradient method to find a viable control
  $f(x) = \sum_{i=1}^{n} \alpha_i y_i k(x_i, x) + b$



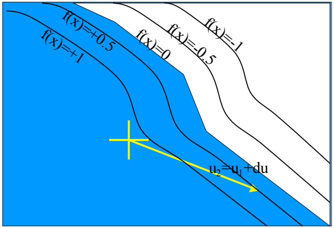
Iteration $n+1$

- Gradient method to find a viable control $f(x) = \sum_{i=1}^{n} \alpha_i y_i k(x_i, x) + b$



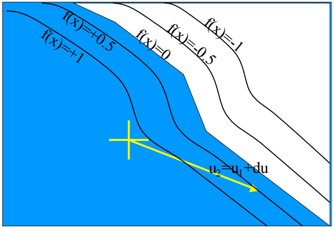
Iteration $n+1$

- Gradient method to find a viable control
  $f(x) = \sum_{i=1}^{n} \alpha_i y_i k(x_i, x) + b$



Iteration $n+1$

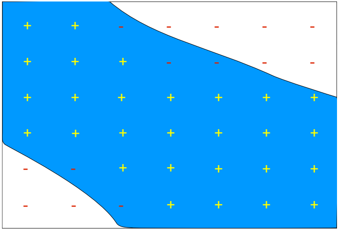
- Gradient method to find a viable control
  $f(x) = \sum_{i=1}^{n} \alpha_i y_i k(x_i, x) + b$



- Possible to extend to several time steps

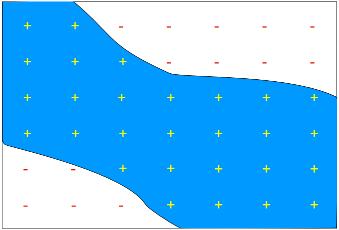
Iteration $n+1$

- Update of the labels from SVM_n



Iteration $n+1$

- Define SVM_{n+1}



- Theorem: Under general assomptions on the quality of learning and on function φ, the algorithm provides an approximation of the viability kernel which converges to the actual viability kernel when the resolution of the grid tends to 0

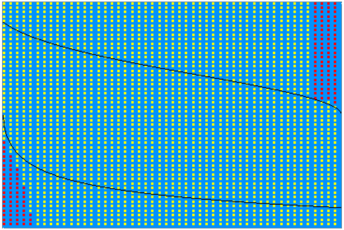
- Simplified model of the growth of a population in a limited space
- Dynamical system

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

- Under constraints
  - $x \in [a, b]$
  - $y \in [d, e]$
  - $u \in [-c, c]$

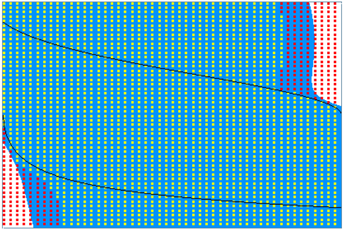
Progressive approximation of the viability kernel

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



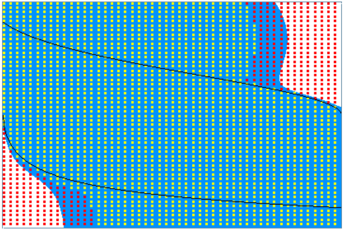
Progressive approximation of the viability kernel

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



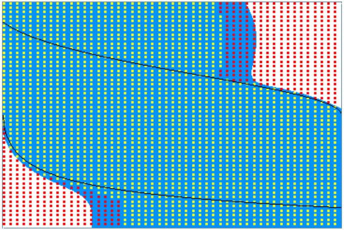
Progressive approximation of the viability kernel

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



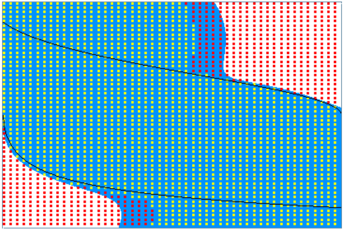
Progressive approximation of the viability kernel

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



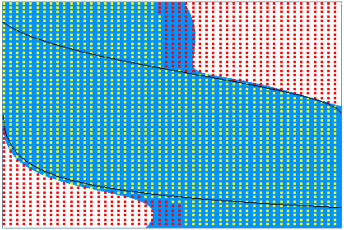
Progressive approximation of the viability kernel

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



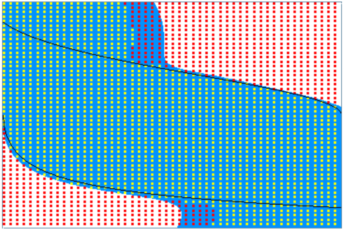
Progressive approximation of the viability kernel

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



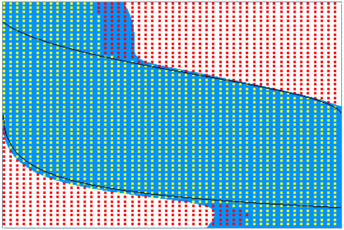
Progressive approximation of the viability kernel

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



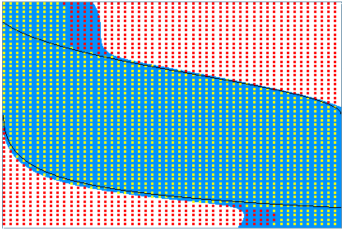
Progressive approximation of the viability kernel

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



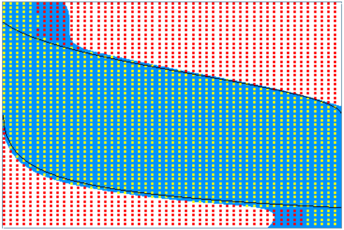
Progressive approximation of the viability kernel

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



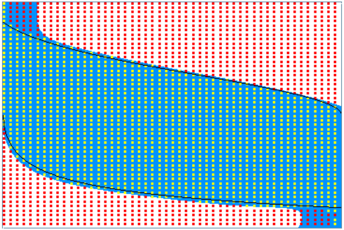
Progressive approximation of the viability kernel

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



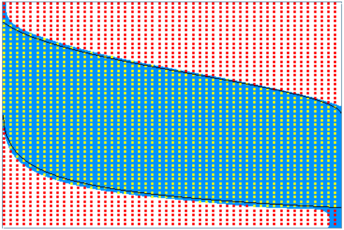
Progressive approximation of the viability kernel

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



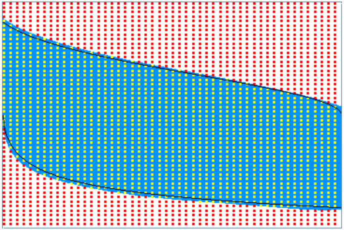
Progressive approximation of the viability kernel

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



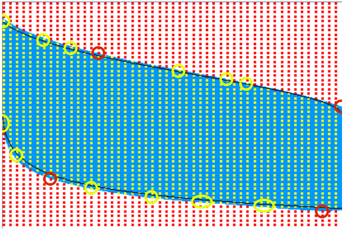
Progressive approximation of the viability kernel

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



Progressive approximation of the viability kernel

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

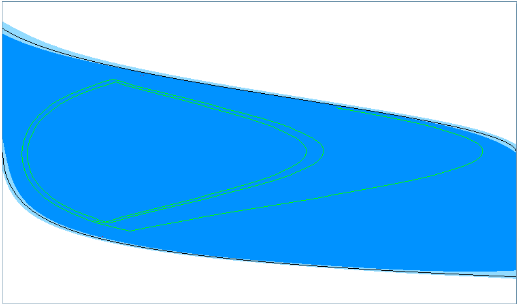


- 12 iterations, 19 SV

SVM Heavy Controller

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

Example of controller (5 time steps anticipation)

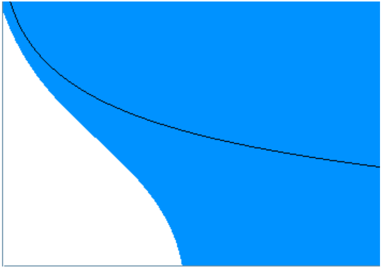


Outline

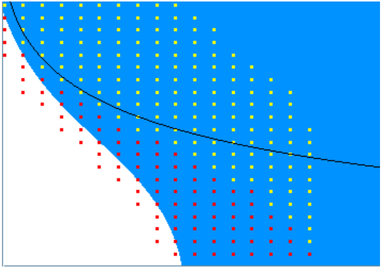
1. Viability theory
2. SVM viability controller
3. SVM viability controller active learning
4. Discussion and perspectives

- The previous algorithm allows to work with control of high dimension
- But what about the dimension of the state space?
- Active learning: limits the number of points to label / to use for SVM training
  - labeling instances is time consuming
  - the size of the grid is exponential with the dimension
  - training the SVM is roughly quadratic with the training sample size

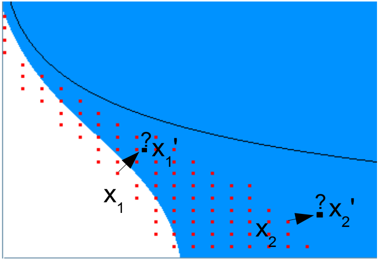
- Starting from a given SVM



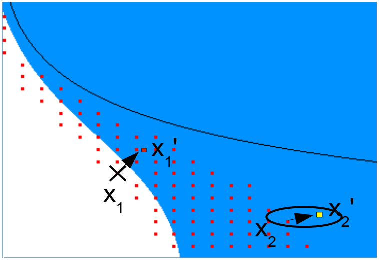
- Test the points that are likely to leave



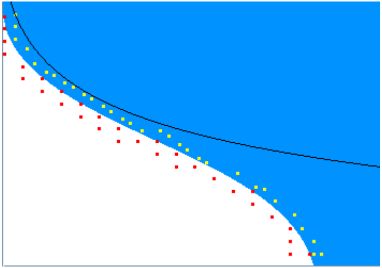
- Are they -1 and near +1?



- Are they -1 and near +1?

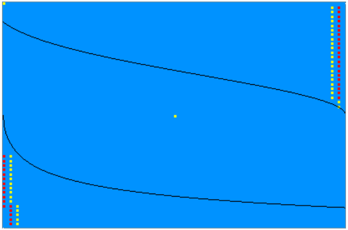


- Keep a -1 (on the grid) and a +1



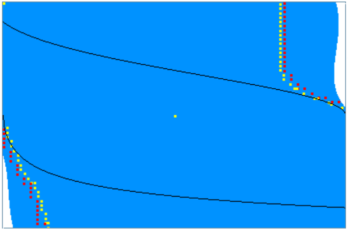
Progressive approximation of the viability kernel

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



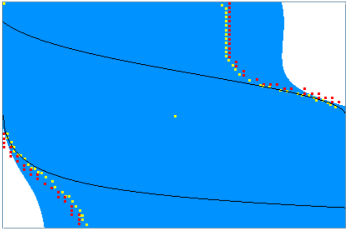
Progressive approximation of the viability kernel

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



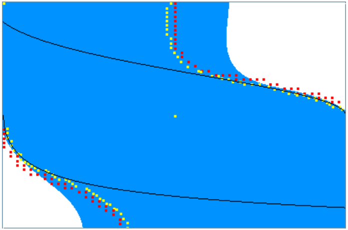
Progressive approximation of the viability kernel

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



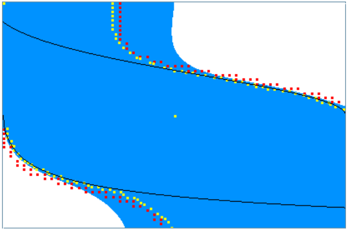
Progressive approximation of the viability kernel

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



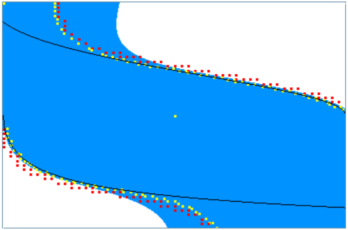
Progressive approximation of the viability kernel

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



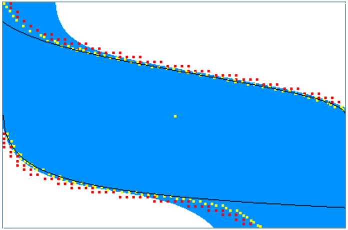
Progressive approximation of the viability kernel

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



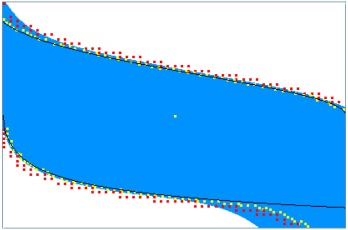
Progressive approximation of the viability kernel

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



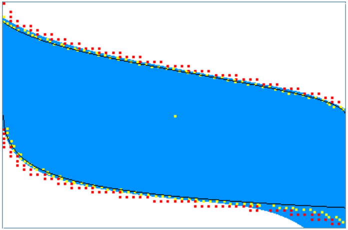
Progressive approximation of the viability kernel

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



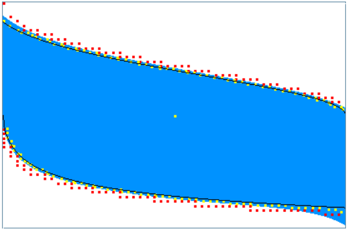
Progressive approximation of the viability kernel

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



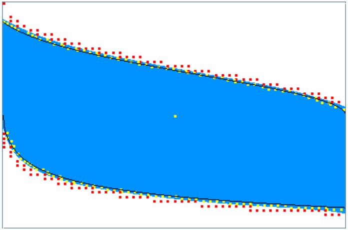
Progressive approximation of the viability kernel

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



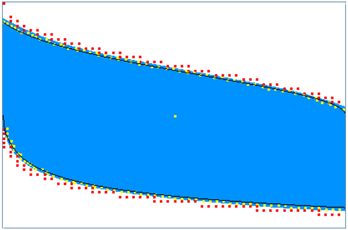
Progressive approximation of the viability kernel

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



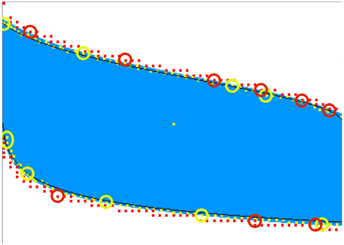
Progressive approximation of the viability kernel

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



Progressive approximation of the viability kernel

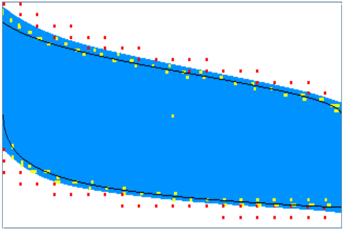
- State space in 2 dimensions, grid of 2601 points, 6 time steps



- 12 iterations, 19 SV, 11% of the grid to compute the SVM

Extending the state space

- State space in 4 dimensions, grid of $\approx$ 200 000 points, 4 time steps



- 14 iterations, 347 SV, 26% of the grid to compute the SVM

Outline

Discussion and perspectives

Advantages of using SVMs to approximate viability kernels:

- Enable to use gradient techniques to find viable controls, which is more efficient than systematic search
- Provide easily more or less cautious controllers

Active learning allows to decrease of one dimension the number of SVM training examples

Perspectives

- More efficient active learning techniques should decrease more significantly training samples size
- Goal: Use training set of size similar to the number of SV