SVM viability controller active learning
Application to bike control

Laetitia Chapel   Guillaume Deffuant

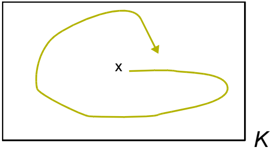
Laboratory of Engineering for Complex Systems (LISC)

IEEE ADPRL, April 2007

Introduction

- Control a dynamical system such that it can survive inside a given set of admissible states (and possibly reach a target)
- State $x(t)$, controls $u(t)$, in discrete time

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



Problem: control a system in order to keep it inside K

- Control a population such that it stays inside a given interval
- Drive a bike on a track
- Drive a car such that it can reach the top of the hill

Outline

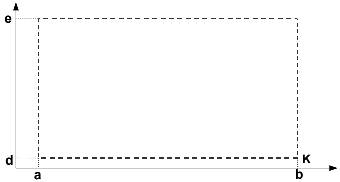
1. Population problem
2. Bike on a track
3. Car on the hill

Outline

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

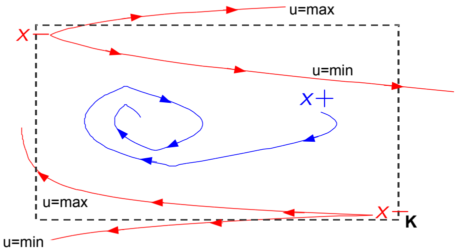
$$\begin{cases} x(t+dt) = x(t) + x(t)y(t)dt \\ y(t+dt) = y(t) + u(t)dt \end{cases} \quad (2)$$

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



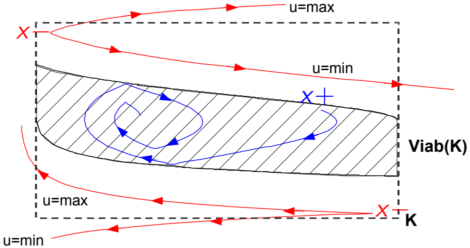
How controlling the system such that it always stays in K?

- Dynamic programming approach
  - Coquelin, Martin & Munos, *A dynamic programming approach to viability problem*, ADPRL07
- SVM viability controller



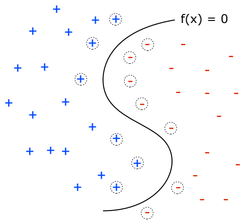
How controlling the system such that it always stays in K?

- Dynamic programming approach
  - Coquelin, Martin & Munos, *A dynamic programming approach to viability problem*, ADPRL07
- SVM viability controller



Support Vector Machines

- Separating hyperplane in a feature space
- $f(x) = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b$ with $\alpha_i > 0$ SV
- SVM function: function such that $f(x) = 0$

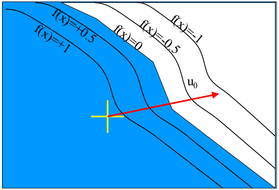


Population problem

- First task: approximate the viability kernel of the system
  - SVM viability algorithm, based on the discretization of K
  - Use of active learning techniques to work in higher dimensional spaces
- Second tack: using SVM function to control the system

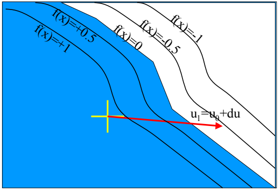
SVM viability kernel approximation

- Iterative algorithm: points of the grid viable at the next step $\rightarrow$ label +1 the others $\rightarrow$ label -1
- SVM function provides a kind of barrier function on the viability kernel boundary
- How determine the label of the points? Gradient method to find a viable control



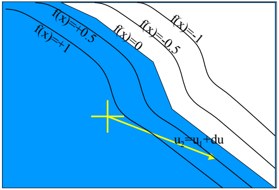
SVM viability kernel approximation

- Iterative algorithm: points of the grid viable at the next step $\rightarrow$ label +1 the others $\rightarrow$ label -1
- SVM function provides a kind of barrier function on the viability kernel boundary
- How determine the label of the points? Gradient method to find a viable control



SVM viability kernel approximation

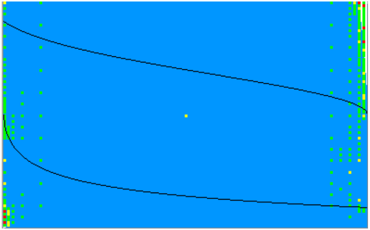
- Iterative algorithm: points of the grid viable at the next step $\rightarrow$ label $+1$ the others $\rightarrow$ label -1
- SVM function provides a kind of barrier function on the viability kernel boundary
- How determine the label of the points? Gradient method to find a viable control



SVM viability controller active learning

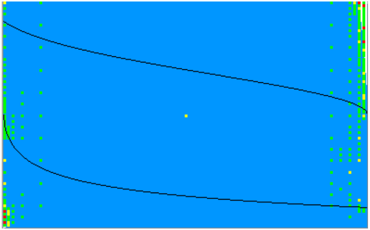
- Combine an adaptive grid and active learning procedure
- Active learning: limits the number of points to use for SVM training
  - the size of the grid is exponential with the dimension
  - training the SVM is roughly quadratic with the training sample size
- Aim: use a number of points near the number of SV
- Which points to choose? We focus on the boundary

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



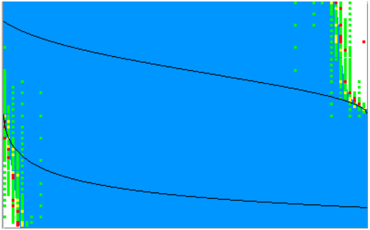
- 28 SV, 124 (2%) max in S

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



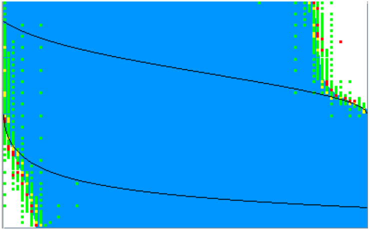
- 28 SV, 124 (2%) max in S

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



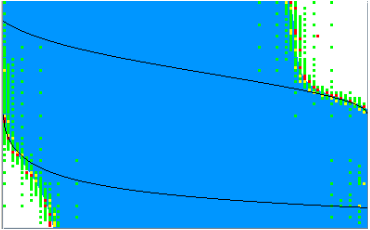
- 28 SV, 124 (2%) max in S

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



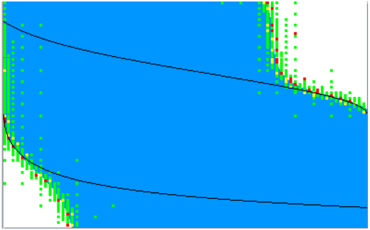
- 28 SV, 124 (2%) max in S

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



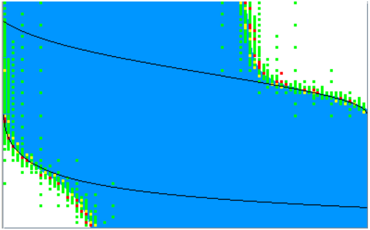
- 28 SV, 124 (2%) max in S

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



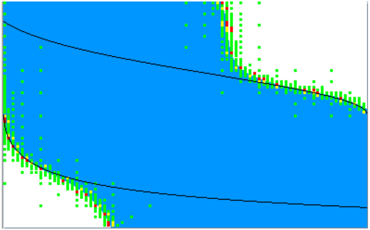
- 28 SV, 124 (2%) max in S

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



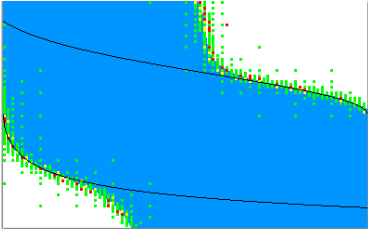
- 28 SV, 124 (2%) max in S

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



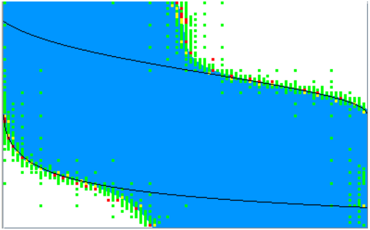
- 28 SV, 124 (2%) max in S

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



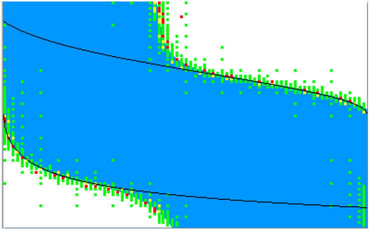
- 28 SV, 124 (2%) max in S

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



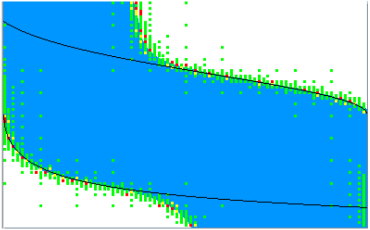
- 28 SV, 124 (2%) max in S

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



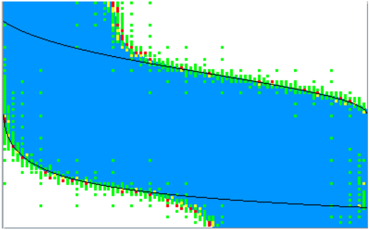
- 28 SV, 124 (2%) max in S

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



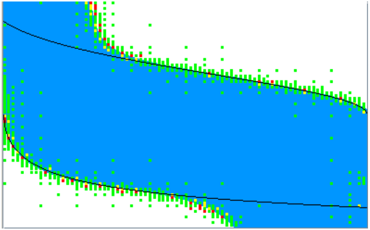
- 28 SV, 124 (2%) max in S

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



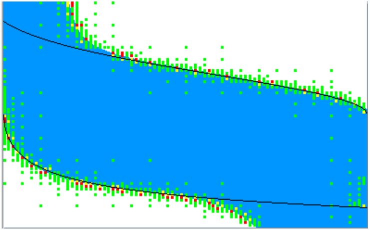
- 28 SV, 124 (2%) max in S

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



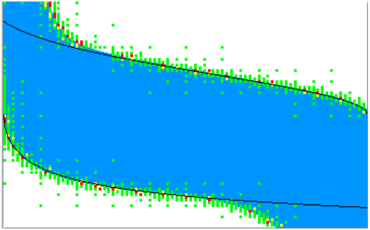
- 28 SV, 124 (2%) max in S

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



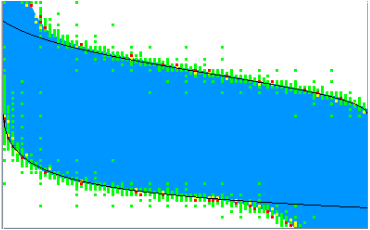
- 28 SV, 124 (2%) max in S

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



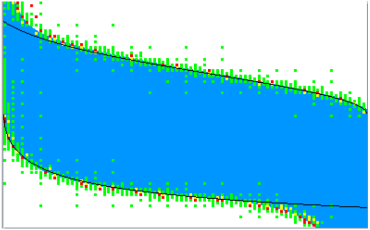
- 28 SV, 124 (2%) max in S

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



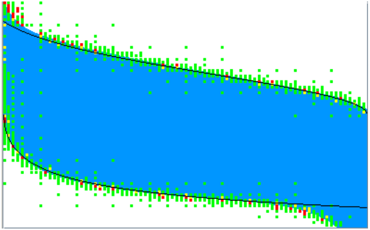
- 28 SV, 124 (2%) max in S

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



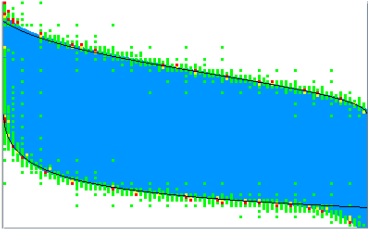
- 28 SV, 124 (2%) max in S

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



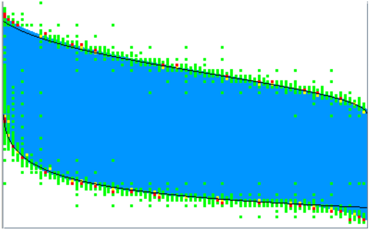
- 28 SV, 124 (2%) max in S

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



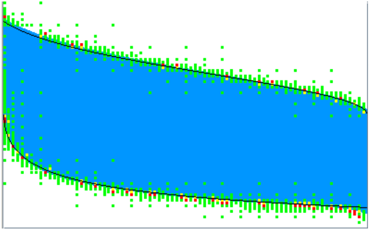
- 28 SV, 124 (2%) max in S

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



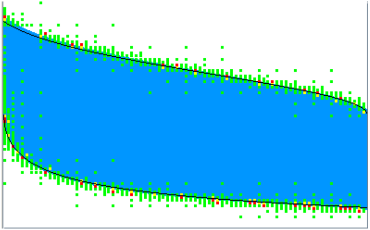
- 28 SV, 124 (2%) max in S

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



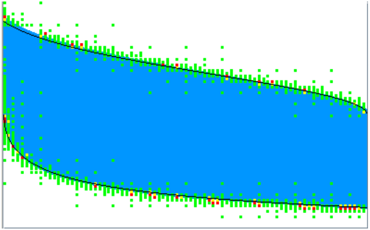
- 28 SV, 124 (2%) max in S

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



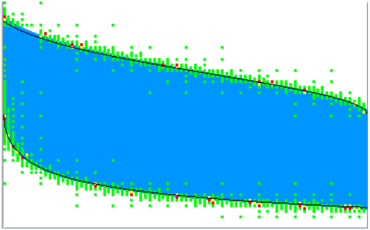
- 28 SV, 124 (2%) max in S

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



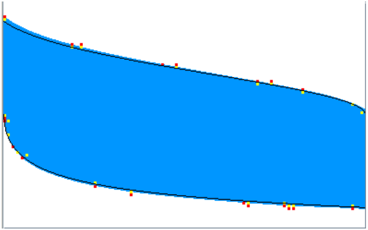
- 28 SV, 124 (2%) max in S

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



- 28 SV, 124 (2%) max in S

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

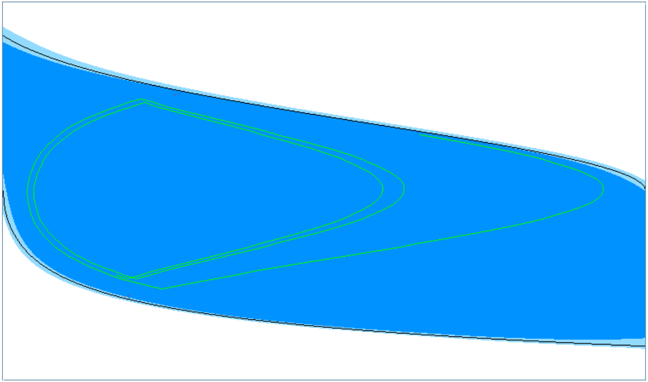


- 28 SV, 124 (2%) max in S

SVM Heavy Controller $\neq$ optimal 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)

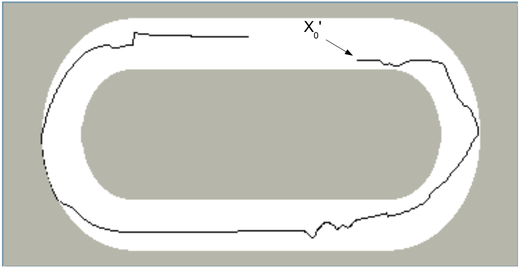


Outline

- Randlov: drive a bike to a target
- 6-dimensional system
  - angle the handlebars are displaced from vertical and velocity of the angle
  - angle from the bicycle to vertical and its velocity
  - position of the front wheel and angle of tilt of the bike
- 2 control variables
  - torque applied to the handlebars
  - displacement of the bike
- Constraints on the states

- First task: approximate the viability kernel in dimension 6
  - 531441 points on the whole grid
  - 3914 SV, 34028 (6.5%) max in S
- Second task: control the system
  - Aim: driving a bike in a track without going outside and without falling

- Step 2: control on a 2d track

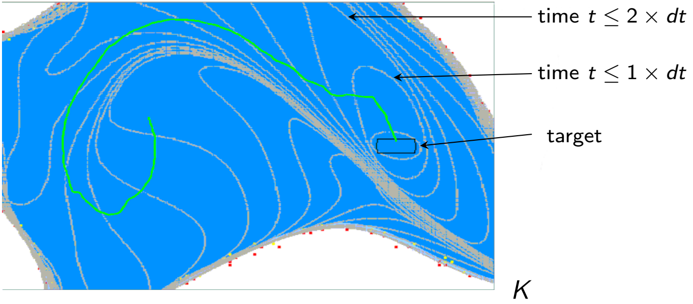


Outline

1. Population problem
2. Bike on a track
3. Car on the hill

Car on the hill

- Car on the hill: the car much reach the top of the hill, without falling
- State space in 2 dimensions, 1 dimensional control



Conclusion

- Viability theory: control a system to maintain it inside K (and possibly reach a target)
- SVM allow to use active learning techniques

But

- Based on the distance of the points to the SVM boundary, which is not direct to compute $\longrightarrow$ greedy in time