Approximate planning, controller synthesis, and anomaly detection through variational inequalities and linear complementarity problems

Erik Zawadzki epz@cs.cmu.edu

André Platzer aplatzer@cs.cmu.edu

Geoffrey Gordon ggordon@cs.cmu.edu

Variational inequalities (VIs), linear complementarity problems (LCPs), and approximation

What are variational inequalities?

- Variational inequalities represent important conditions that occur frequently in equilibrium and optimization problems.
  - Based on an operator *F* and a subset *C:*

$$\langle y - x, Fx \rangle \geq 0, \forall y \in C$$

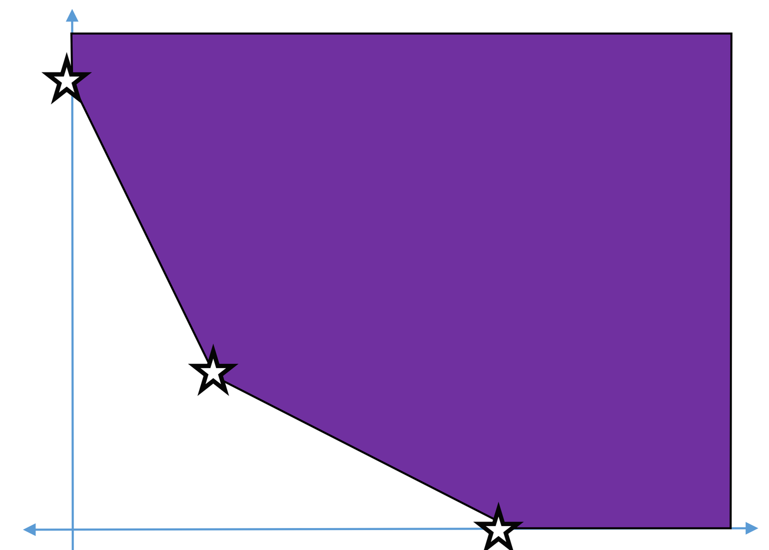
  - First order sufficient conditions for minimizing a proper convex function *f* over a convex set *C* is an important example

$$\langle y - x, \partial f(x) \rangle \geq 0, \forall y \in C$$

    - Any feasible direction $y - x$ must have a non-negative first-order variation, increasing the function value
  - Application areas include:
    - Optimization; in particular this encompasses planning, reinforcement learning, control synthesis, classification, and anomaly detection
    - Equilibria finding in engineering including traffic equilibria and structural equilibria
    - Equilibria finding in game theory; sufficient conditions for the Nash equilibrium can be expressed as a variational inequality
    - Physical simulation; especially those involving non-penetration constraints
- LCPs are a restricted class of VIs where the operator is affine and the subset is a cone

$$K \ni x \perp y = Mx + q \geq x \in K^*$$

  - The KKT system for quadratic programs are LCPs:



Plot of the three solution for LCP $M = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}, q = -\begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$$\min_x x^\top Qx + c^\top x \quad \text{Subject to} \quad Ax \leq b, \; Cx \leq d \iff \mathcal{L}(x, \lambda, \mu) = x^\top Qx + c^\top x + \lambda^\top(Ax - b) + \mu^\top(Cx - d) \iff \begin{bmatrix} \mathbb{R}^q \\ \mathbb{R}^r_+ \\ \mathbb{R}^s \end{bmatrix} \ni \begin{bmatrix} x \\ \lambda \\ \mu \end{bmatrix} \perp \begin{bmatrix} Q & A^\top & C \\ -A & 0 & 0 \\ -C & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \\ \mu \end{bmatrix} + \begin{bmatrix} c \\ b \\ d \end{bmatrix} \in \begin{bmatrix} 0 \\ \mathbb{R}^r_+ \\ 0 \end{bmatrix}$$

What is this project about?

- Our work is on solving monotone LCPs approximately, by solving a ***projective LCP*** that approximates the above system within the span of some basis Φ
  - An operator *F* is monotone over set *C* if

$$\langle Fx - Fy, x - y \rangle \geq 0, \forall x, y \in C$$

  - Monotone operators are related to convex problems: the KKT operator associated with convex problems are monotone

$$\mathrm{N} = \Pi_\Phi M + \Pi_\perp, \quad r = \Pi_\Phi \mathrm{q}, \quad 0 \leq x \perp \Pi_\Phi Mx + \Pi_\perp x + \Pi_\Phi q \geq 0$$

  - Developed a fast interior point solver that works with projective LCPs in time that is $O(nk^2)$ per iteration, rather than $O(n^{2+\epsilon})$, a huge saving if $k \ll n$

Application: classification via support vector machines (SVMs)

- Support vector machines (SVMs) are a machine learning model useful for classification and regression
- SVMs find hyperplanes (in some features space) that do a good job of splitting positively labeled data from negatively labeled data
  - Using different feature spaces leads to decision boundaries that can appear highly non-linear
- Fitting the (hard margin) SVM model can be done with a quadratic program that maximizes the margin between positive and negative points.
  - Y is a diagonal matrix of the labels
  - X be an $N \times d$ feature matrix.
  - w is the normal vector; want a normal vector with small magnitude

$$\min_w \tfrac{1}{2}\|w\|_2^2 \quad \text{Subject to} \quad YXw \geq 1$$

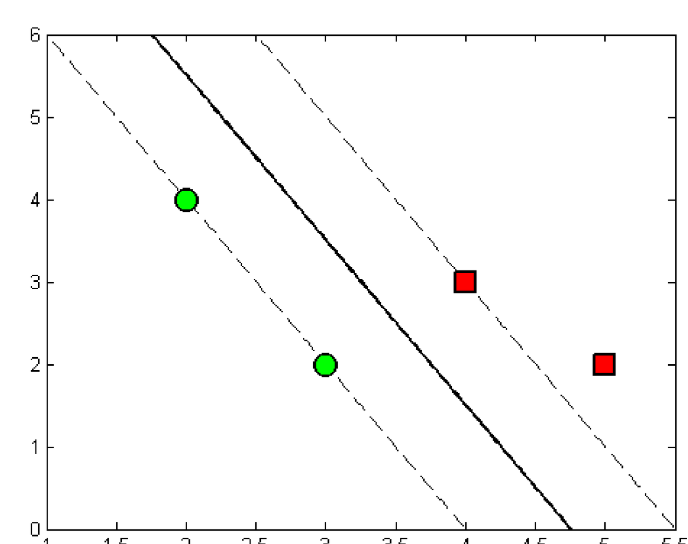
- The dual program, after some rearrangement, is a simply constrained cone-constrained problem:

$$\min_{\alpha \geq 0} \alpha^\top YXX^\top Y\alpha - 1^\top \alpha$$

  - $y \circ \alpha$ is the weight for each point in a decision function:

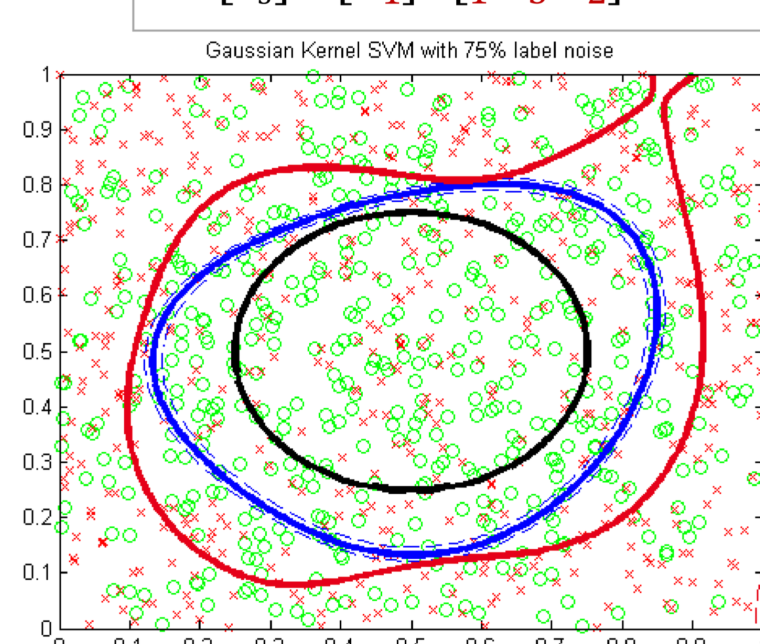
$$f(x) = \mathrm{sgn}[\textstyle\sum_i^n \alpha_i y_i k(x, x_i)]$$

- This can be simply written as an LCP where $\mathrm{M} = YXX^\top Y$ and $q = -1$
- Our approach approximates the symmetric, monotone SVM LCP using a projective LPC
- Good approximation tends to smooth out the α
  - Exact SVMs have sparse α; points corresponding to non-zero components of α are called *support vectors*
  - This smoothing behavior seems to have nice statistical properties
    - Limits how precisely a single point's weights can be set
    - Approximation can make the SVM fit more robust to label noise



Simple four point SVM example. Illustrates the margin, and the SVM LCP.

$$0 \leq \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \perp \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \circ \begin{bmatrix} 1 & 2 & 4 \\ 1 & 3 & 2 \\ 1 & 4 & 3 \\ 1 & 5 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 3 & 4 & 5 \\ 4 & 2 & 3 & 2 \end{bmatrix} \circ \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} - 1 \geq 0$$

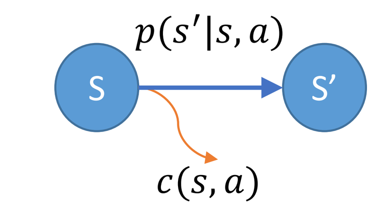


Our method fits the blue curve 200x faster than the looser red curve. Black is ground truth.

1500 examples; Gaussian kernel; Chebyshev basis.

Application: policy synthesis via Markov Decision Processes (MDPs)

- MDPs are a general framework for sequential decision making under uncertainty
- A solution is a *policy*, or a function that maps states to optimal actions
  - Finding a policy called planning, or controller synthesis, or learning
- Discrete states transition to other discrete states probabilistically
  - *Transition function* $p(s'|s,a)$ governs how states transition to a new state s' given an action
  - *Cost function* $c(s,a)$ describes that immediate cost of a state-action pair
    - Costly states may still have high *value*—paying an initial up-front cost might be optimal



- Exact problem can be solved with iterative procedures like *value iteration* or directly with a linear program
  - Value iteration finds the value function $v(s) = \min_a [c(s,a) + \gamma \mathbb{E}_{s'} v(s')]$
    - The value function describes the long-term cost of being in a particular state
    - Implicitly describes the optimal policy: $\pi^*(s) = \mathrm{argmin}_a [c(s,a) + \gamma \mathbb{E}_{s'} v(s')]$
  - MDPs can be solved via the following linear programs :

$$\min_v -p^\top v \quad \text{Subject to} \quad v \leq c_a + \gamma P_a^\top v, \forall a \in A \quad \xleftrightarrow{\text{Dual to}} \quad \min_v \sum_a^A c_a f_a \quad \text{Subject to} \quad \textstyle\sum_a^A f_a = p + \sum_a^A P_a f_a$$

  - The dual variables are *flow function* that that describe the expected number of times the system will perform action *a* in state *s:* $f_a(s) = \sum_{t=0}^\infty \gamma^t P(s_t = s, a_t = a)$
    - Also implicitly describe the optimal policy: $\pi^*(s) = \mathrm{argmax}_a f_a(s)$
  - All linear programs can be expressed as LCPs; this is the MDP LCP:

$$\begin{bmatrix} \mathbb{R}^S \\ \mathbb{R}^S_+ \\ \mathbb{R}^S_+ \end{bmatrix} \ni \begin{bmatrix} v \\ f_1 \\ f_2 \end{bmatrix} \perp \begin{bmatrix} 0 & I - \gamma P_1 & I - \gamma P_2 \\ \gamma P_1^\top - I & 0 & 0 \\ \gamma P_2^\top - I & 0 & 0 \end{bmatrix} \begin{bmatrix} v \\ f_1 \\ f_2 \end{bmatrix} + \begin{bmatrix} -p \\ c_1 \\ c_2 \end{bmatrix} \in \begin{bmatrix} 0 \\ \mathbb{R}^S_+ \\ \mathbb{R}^S_+ \end{bmatrix}$$

- General purpose MDP solvers work on discrete state-spaces
  - Can model continuous physical systems (e.g. robotic control, plant control) by discretizing dynamics
- State-space may be ***huge;*** many important MDPs are intractable to solve exactly
  - *D*-dimensional continuous physical system with *N* points per dimension has N^D states.
  - This is usually intractable for $N \geq 5$ (depends on smoothness of dynamics)
  - Called the "*curse of dimensionality"*
- Approximation solution methods, like least-squares policy iteration, policy search, and fitted value iteration are necessary to tackle many real-world planning problems
- Our approach approximations both the value and flow functions via a projective LCP
  - Can use both approximate value and flow functions together via an actor-critic RL method, like Monte Carlo Tree Search