Learning RVO

David Millman

December 10, 2007

The basic question

Can we use Machine Learning to accelerate aspects of motion planning such as collision avoidance or path planning?

Review RVO

- Calculate preferred velocity
- Find neighbors
- Compute new velocities

Support Vector Regression, $\epsilon-$SVR

- Training data: $\{(x_1, y_1), \ldots, (x_n, y_n)\} \subset \mathcal{X} \times \mathbb{R}$
- Goal: find function $f(x)$ which has at most ϵ deviation from actual targets y_i and is as *flat* as possible
- Errors
  - OK: Less then ϵ
  - NOT OK: Greater then ϵ
- Motion planning example, pick a direction which is ϵ close to the desired direction

Basic idea with Linear Kernel

- Linear function

$$f(x) = \langle w, x \rangle + b \text{ with } w \in \mathcal{X}, b \in \mathbb{R}$$

- Denote $\langle \cdot\,, \cdot \rangle$ as the dot product in $\mathcal{X}$
- What is *flat*?

Basic idea with Linear Kernel

- Linear function

$$f(x) = \langle w, x \rangle + b \text{ with } w \in \mathcal{X}, b \in \mathbb{R}$$

- Denote $\langle \cdot\,, \cdot \rangle$ as the dot product in $\mathcal{X}$
- What is *flat*?
  - Minimize w!

Basic idea with Linear Kernel

- Linear function

$$f(x) = \langle w, x\rangle + b \text{ with } w \in \mathcal{X}, b \in \mathbb{R}$$

- Denote $\langle \cdot\, , \cdot \rangle$ as the dot product in $\mathcal{X}$
- What is *flat*?
  - Minimize w! $\implies$ Minimize norm, $\|w\|^2 = \langle w, w\rangle$

Basic idea with Linear Kernel

- Linear function

$$f(x) = \langle w, x \rangle + b \text{ with } w \in \mathcal{X}, b \in \mathbb{R}$$

- Denote $\langle \cdot , \cdot \rangle$ as the dot product in $\mathcal{X}$
- What is *flat*?
  - Minimize w! $\implies$ Minimize norm, $\|w\|^2 = \langle w, w \rangle$

Optimization problem

$$\text{minimize} \quad \frac{1}{2}\|w\|^2$$

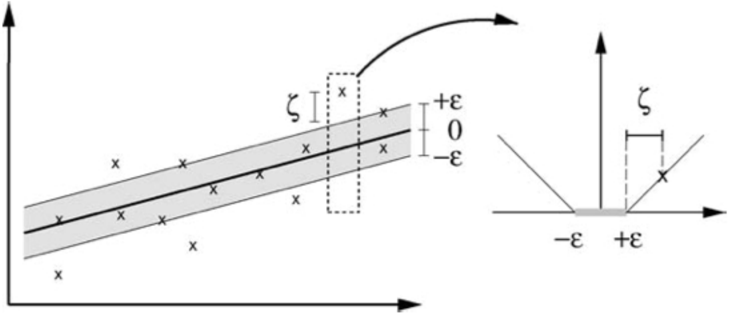
$$\text{subject to} \quad \begin{cases} y_i - \langle w, x_i \rangle - b & \leq & \epsilon \\ \langle w, x_i \rangle + b - y_i & \leq & \epsilon \end{cases}$$

Basic idea with Linear Kernel

- Assumes this is possible
- But what if it does not?

Basic idea with Linear Kernel

- Assumes this is possible
- But what if it does not?
    - Slack variables ξ_i, ξ_i^*



Basic idea with Linear Kernel

- Resulting in the optimization problem

$$\text{minimize} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$

$$\text{subject to} \quad \begin{cases} y_i - \langle w, x_i \rangle - b & \leq & \epsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i & \leq & \epsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq & 0 \end{cases}$$

$$\text{where} \quad |\xi_\epsilon| = \begin{cases} 0 & \text{if } |\xi| < \epsilon \\ |\xi| - \epsilon & \text{otherwise} \end{cases}$$

Basic idea with Linear Kernel

- Take it to the duel

$$\text{maximize} \quad \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle \\ -\epsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*) \end{cases}$$

$$\text{subject to} \quad \sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C]$$

where α are Lagrange multipliers

Basic idea with Linear Kernel

- And finally *Support Vector expansion*, w written as a linear combination of x_i

$$
\begin{aligned}
w &= \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) x_i \Longrightarrow \\
f(x) &= \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) \langle x_i, x \rangle + b
\end{aligned}
$$

RVO and Leaning

- Calculate preferred velocity, v_{pref}
- Find neighbors N
- Compute new velocities
  (1) Input, v_{pref}, N
  (2) Sample 200 points in set of admissible new velocities for agent A_i at velocity v_i
  (3) Select velocity with minimum penalty

RVO and Leaning

- Calculate preferred velocity, v_{pref}
- Find neighbors N
- Compute new velocities
  (1) Input, v_{pref}, N
  (2) Sample 200 points in set of admissible new velocities for agent A_i at velocity v_i
  (3) Select velocity with minimum penalty

Can (2) be learned? If so can it be faster?

Results, A general explanation

Table: Linear Kernel with 2 agents, 600 data points

C	ϵ	MSE	SCC
1	0.25	0.0355369	0.0596180
1	0.75	0.0796566	0.0311745
2	0.25	0.0355341	0.0596206
2	0.75	0.0827220	0.0307352
4	0.25	0.0355332	0.0596217
4	0.75	0.0834387	0.0304184
8	0.25	0.0355465	0.0594722
8	0.75	0.0834401	0.0304220
16	0.25	0.0355456	0.0595128
16	0.75	0.0834508	0.0304255
32	0.25	0.0355425	0.0595311

Results, Varying ϵ

Table: Linear Kernel with 2 agents, 600 data points

C	ϵ	MSE	SCC
1	0.05	0.0352513	0.0558446
1	0.10	0.0350025	0.0599738
1	0.15	0.0348204	0.0627678
1	0.20	0.0350328	0.0609245
1	0.25	0.0355369	0.0596180
2	0.05	0.0352517	0.0558258
2	0.10	0.0350038	0.0599357
2	0.15	0.0348187	0.0628222
2	0.20	0.0350312	0.0609415
2	0.25	0.0355341	0.0596206

Results, Varying number of agents

Table: Linear Kernel with 4 agents, 320 data points

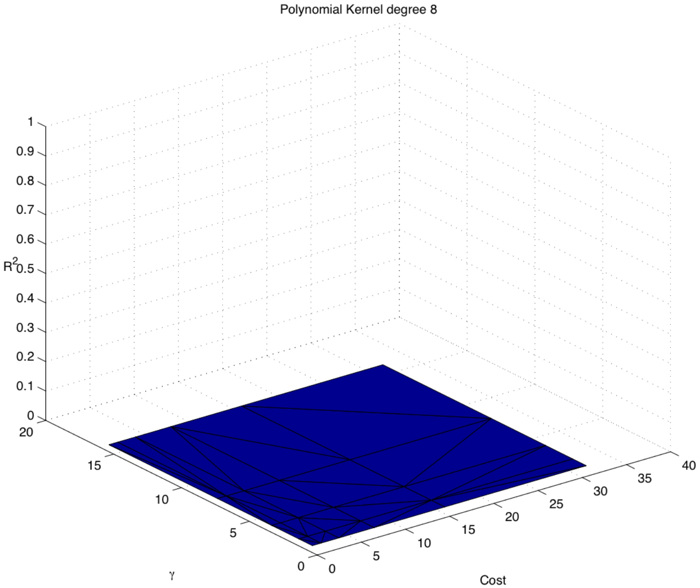
C	ϵ	MSE	SCC
1	0.25	0.0464268	0.029539000
1	0.75	0.0516781	0.000299641
2	0.25	0.0479181	0.031862200
2	0.75	0.0516781	0.000299641
4	0.25	0.0483595	0.032727100
4	0.75	0.0516781	0.000299641
8	0.25	0.0483510	0.032774400
8	0.75	0.0516781	0.000299641
16	0.25	0.0483519	0.032738900
16	0.75	0.0516781	0.000299641
32	0.25	0.0483856	0.032189800
32	0.75	0.0516781	0.000299641

Results, Varying number of agents cont.

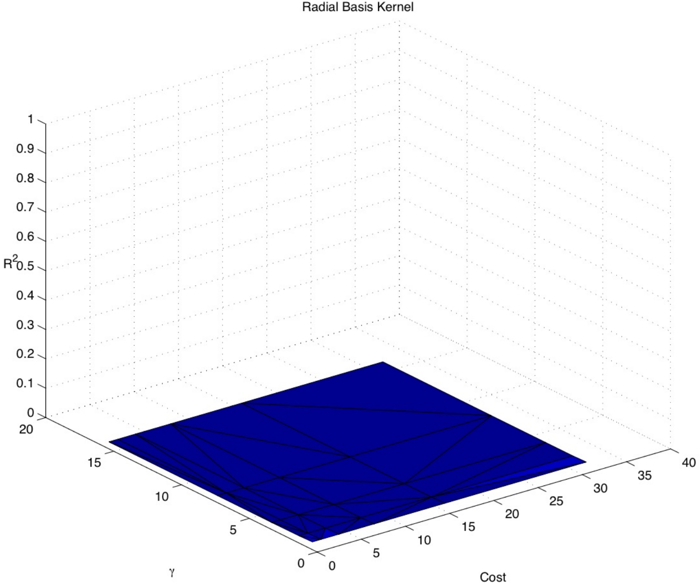
Table: Linear Kernel with 8 agents, 320 data points

C	ϵ	MSE	SCC
1	0.25	0.0472112	0.0360096
1	0.75	0.0513404	0.0006994
2	0.25	0.0474242	0.0352149
2	0.75	0.0513404	0.0006994
4	0.25	0.0479073	0.0350223
4	0.75	0.0513404	0.0006994
8	0.25	0.0481368	0.0343743
8	0.75	0.0513404	0.0006994
16	0.25	0.0484407	0.0351923
16	0.75	0.0513404	0.0006994
32	0.25	0.0488201	0.0345883
32	0.75	0.0513404	0.0006994

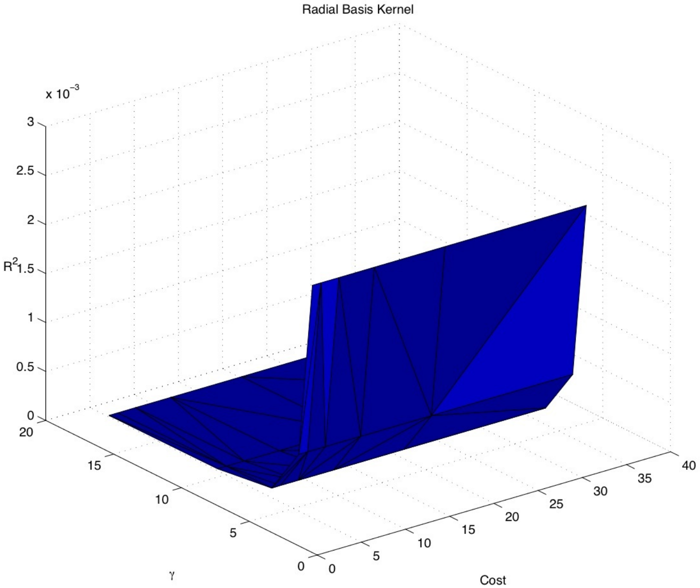
Results, Polynomial Kernel degree 8



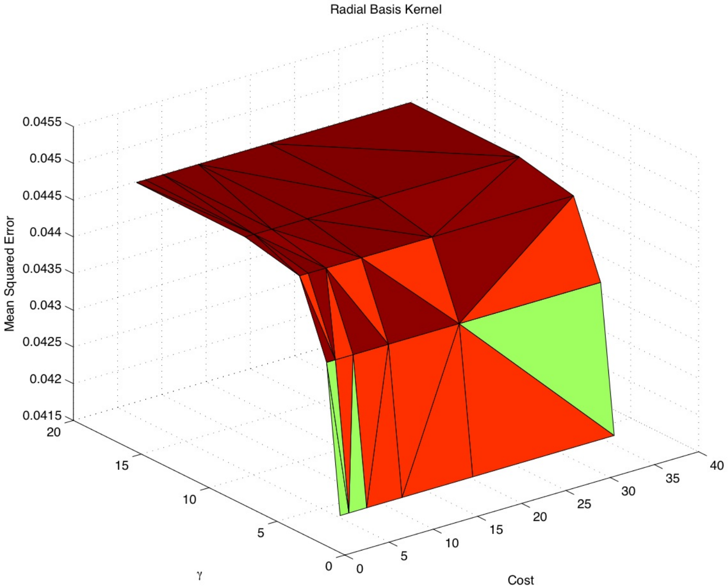
Results, Radial Basis Kernel



Results, Radial Basis Kernel SCC



Results, Radials Basis Kernel MSE

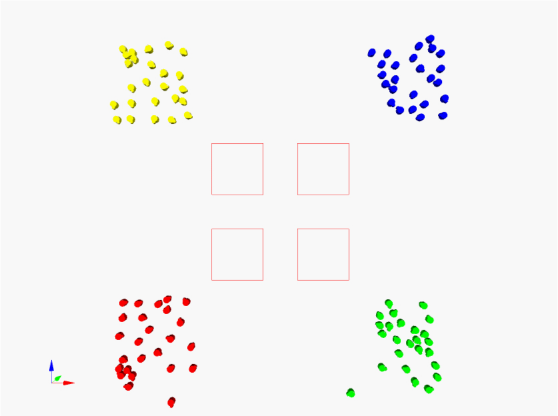


Results, Timings with model complexity

Table: Time per eval

Eval type	Time (sec)
Orig	0.00012487223
Large Model	0.0015907775
Med Model	4.9171695e-05
Small Model	5.14325595e-06

Video



One month

- features determined by radius and number neighbors
- basic regression
- scaling in RVO
- RVO uniform sampeling

Six months

- Other ML methods
  - Bivariate SVR
  - Bivariate regression
  - Neuro-Net
- Learn in polar coordinates
- Desperation Planning - a probabilistic complete version of Viability Filtering
  - Implementation
  - Compare with RRT-Blossom with VF

B. Blumberg, M. Downie, Y. Ivanov, M. Berlin, M. P. Johnson, and B. Tomlinson.
Integrated learning for interactive synthetic characters.
In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 417–426, New York, NY, USA, 2002. ACM Press.

T. Conde and D. Thalmann.
Learnable behavioural model for autonomous virtual agents: low-level learning.
In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 89–96, New York, NY, USA, 2006. ACM Press.

References II

M. L. Jur van den Berg and D. Manocha.
Reciprocal velocity obstacles for real-time multi-agent collision avoidance.
Technical report, Department of Computer Science, UNC, 2007.

M. Kalisiak and M. van de Panne.
Faster motion planning using learned local viability models.
In *ICRA*, pages 2700–2705, 2007.

J. Miura.
Support vector path planning.
In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2894–2899, 2006.

References III

A. Y. Ng and S. Russell.
Algorithms for inverse reinforcement learning.
In *Proc. 17th International Conf. on Machine Learning*, pages 663–670. Morgan Kaufmann, San Francisco, CA, 2000.

S. J. Russell and A. Zimdars.
Q-decomposition for reinforcement learning agents.
In *ICML*, pages 656–663, 2003.

A. Smola and B. Schoelkopf.
A tutorial on support vector regression, 1998.