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?
- Calculate preferred velocity
- Find neighbors
- Compute new velocities
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 \( \epsilon \) deviation from actual targets \( y_i \) and is as flat as possible

Errors
- OK: Less then \( \epsilon \)
- NOT OK: Greater then \( \epsilon \)

Motion planning example, pick a direction which is \( \epsilon \) 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
  \[ f(x) = \langle w, x \rangle + b \] with \( w \in X, b \in \mathbb{R} \)
- Denote \( \langle \cdot, \cdot \rangle \) as the dot product in \( X \)
- What is \textit{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 \( w \! = \! \Longleftrightarrow \text{Minimize norm, } \|w\|^2 = \langle w, w \rangle \)

Optimization problem

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|w\|^2 \\
\text{subject to} & \quad \begin{cases} 
  y_i - \langle w, x_i \rangle - b & \leq \epsilon \\
  \langle w, x_i \rangle + b - y_i & \leq \epsilon 
\end{cases}
\end{align*}
\]
Basic idea with Linear Kernel

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

Slack variables $\xi_i, \xi^*_i$
Basic idea with Linear Kernel

- Assumes this is possible
- But what if it does not?
  - Slack variables $\xi_i, \xi_i^*$
Basic idea with Linear Kernel

Resulting in the optimization problem

\[
\text{minimize} \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} (\xi_i + \xi_i^*) \\
\text{subject to} \quad \begin{cases} 
  y_i - \langle w, x_i \rangle - b & \leq \epsilon + \xi_i \\
  \langle w, x_i \rangle + b - y_i & \leq \epsilon + \xi_i^* \\
  \xi_i, \xi_i^* & \geq 0 
\end{cases}
\]

where \( |\xi_\epsilon| = \begin{cases} 
  0 & \text{if } |\xi| < \epsilon \\
  |\xi| - \epsilon & \text{otherwise}
\end{cases} \)
Basic idea with Linear Kernel

- Take it to the duel

\[
\begin{aligned}
&\text{maximize} \\
&\left\{ -\frac{1}{2} \sum_{i,j=1}^{n} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j \rangle \\
&\quad \quad \quad -\epsilon \sum_{i=1}^{n} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i (\alpha_i - \alpha_i^*) \right\} \\
\end{aligned}
\]

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

where \( \alpha \) are Lagrange multipliers
And finally Support Vector expansion, $w$ written as a linear combination of $x_i$

$$
w = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) x_i \implies f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) \langle x_i, x \rangle + b$$
Calculate preferred velocity, $v_{\text{pref}}$

Find neighbors $N$

Compute new velocities

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

Find neighbors \( N \)

Compute new velocities
  (1) Input, \( v_{\text{pref}}, N \)
  (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

<table>
<thead>
<tr>
<th>C</th>
<th>$\epsilon$</th>
<th>MSE</th>
<th>SCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.25</td>
<td>0.0355369</td>
<td>0.0596180</td>
</tr>
<tr>
<td>1</td>
<td>0.75</td>
<td>0.0796566</td>
<td>0.0311745</td>
</tr>
<tr>
<td>2</td>
<td>0.25</td>
<td>0.0355341</td>
<td>0.0596206</td>
</tr>
<tr>
<td>2</td>
<td>0.75</td>
<td>0.0827220</td>
<td>0.0307352</td>
</tr>
<tr>
<td>4</td>
<td>0.25</td>
<td>0.0355332</td>
<td>0.0596217</td>
</tr>
<tr>
<td>4</td>
<td>0.75</td>
<td>0.0834387</td>
<td>0.0304184</td>
</tr>
<tr>
<td>8</td>
<td>0.25</td>
<td>0.0355465</td>
<td>0.0594722</td>
</tr>
<tr>
<td>8</td>
<td>0.75</td>
<td>0.0834401</td>
<td>0.0304220</td>
</tr>
<tr>
<td>16</td>
<td>0.25</td>
<td>0.0355456</td>
<td>0.0595128</td>
</tr>
<tr>
<td>16</td>
<td>0.75</td>
<td>0.0834508</td>
<td>0.0304255</td>
</tr>
<tr>
<td>32</td>
<td>0.25</td>
<td>0.0355425</td>
<td>0.0595311</td>
</tr>
</tbody>
</table>
Table: Linear Kernel with 2 agents, 600 data points

<table>
<thead>
<tr>
<th>C</th>
<th>$\epsilon$</th>
<th>MSE</th>
<th>SCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.05</td>
<td>0.0352513</td>
<td>0.0558446</td>
</tr>
<tr>
<td>1</td>
<td>0.10</td>
<td>0.0350025</td>
<td>0.0599738</td>
</tr>
<tr>
<td>1</td>
<td>0.15</td>
<td>0.0348204</td>
<td>0.0627678</td>
</tr>
<tr>
<td>1</td>
<td>0.20</td>
<td>0.0350328</td>
<td>0.0609245</td>
</tr>
<tr>
<td>1</td>
<td>0.25</td>
<td>0.0355369</td>
<td>0.0596180</td>
</tr>
<tr>
<td>2</td>
<td>0.05</td>
<td>0.0352517</td>
<td>0.0558258</td>
</tr>
<tr>
<td>2</td>
<td>0.10</td>
<td>0.0350038</td>
<td>0.0599357</td>
</tr>
<tr>
<td>2</td>
<td>0.15</td>
<td>0.0348187</td>
<td>0.0628222</td>
</tr>
<tr>
<td>2</td>
<td>0.20</td>
<td>0.0350312</td>
<td>0.0609415</td>
</tr>
<tr>
<td>2</td>
<td>0.25</td>
<td>0.0355341</td>
<td>0.0596206</td>
</tr>
</tbody>
</table>
## Results, Varying number of agents

Table: Linear Kernel with 4 agents, 320 data points

<table>
<thead>
<tr>
<th>C</th>
<th>( \epsilon )</th>
<th>MSE</th>
<th>SCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.25</td>
<td>0.0464268</td>
<td>0.029539000</td>
</tr>
<tr>
<td>1</td>
<td>0.75</td>
<td>0.0516781</td>
<td>0.000299641</td>
</tr>
<tr>
<td>2</td>
<td>0.25</td>
<td>0.0479181</td>
<td>0.031862200</td>
</tr>
<tr>
<td>2</td>
<td>0.75</td>
<td>0.0516781</td>
<td>0.000299641</td>
</tr>
<tr>
<td>4</td>
<td>0.25</td>
<td>0.0483595</td>
<td>0.032727100</td>
</tr>
<tr>
<td>4</td>
<td>0.75</td>
<td>0.0516781</td>
<td>0.000299641</td>
</tr>
<tr>
<td>8</td>
<td>0.25</td>
<td>0.0483510</td>
<td>0.032774400</td>
</tr>
<tr>
<td>8</td>
<td>0.75</td>
<td>0.0516781</td>
<td>0.000299641</td>
</tr>
<tr>
<td>16</td>
<td>0.25</td>
<td>0.0483519</td>
<td>0.032738900</td>
</tr>
<tr>
<td>16</td>
<td>0.75</td>
<td>0.0516781</td>
<td>0.000299641</td>
</tr>
<tr>
<td>32</td>
<td>0.25</td>
<td>0.0483856</td>
<td>0.032189800</td>
</tr>
<tr>
<td>32</td>
<td>0.75</td>
<td>0.0516781</td>
<td>0.000299641</td>
</tr>
</tbody>
</table>
Table: Linear Kernel with 8 agents, 320 data points

<table>
<thead>
<tr>
<th>C</th>
<th>$\epsilon$</th>
<th>MSE</th>
<th>SCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.25</td>
<td>0.0472112</td>
<td>0.0360096</td>
</tr>
<tr>
<td>1</td>
<td>0.75</td>
<td>0.0513404</td>
<td>0.0006994</td>
</tr>
<tr>
<td>2</td>
<td>0.25</td>
<td>0.0474242</td>
<td>0.0352149</td>
</tr>
<tr>
<td>2</td>
<td>0.75</td>
<td>0.0513404</td>
<td>0.0006994</td>
</tr>
<tr>
<td>4</td>
<td>0.25</td>
<td>0.0479073</td>
<td>0.0350223</td>
</tr>
<tr>
<td>4</td>
<td>0.75</td>
<td>0.0513404</td>
<td>0.0006994</td>
</tr>
<tr>
<td>8</td>
<td>0.25</td>
<td>0.0481368</td>
<td>0.0343743</td>
</tr>
<tr>
<td>8</td>
<td>0.75</td>
<td>0.0513404</td>
<td>0.0006994</td>
</tr>
<tr>
<td>16</td>
<td>0.25</td>
<td>0.0484407</td>
<td>0.0351923</td>
</tr>
<tr>
<td>16</td>
<td>0.75</td>
<td>0.0513404</td>
<td>0.0006994</td>
</tr>
<tr>
<td>32</td>
<td>0.25</td>
<td>0.0488201</td>
<td>0.0345883</td>
</tr>
<tr>
<td>32</td>
<td>0.75</td>
<td>0.0513404</td>
<td>0.0006994</td>
</tr>
</tbody>
</table>
Results, Polynomial Kernel degree 8
Results, Radial Basis Kernel

Radial Basis Kernel

$R^2$
Results, Radial Basis Kernel SCC

![Radial Basis Kernel](image)

David Millman
Learning RVO
### Results, Timings with model complexity

#### Table: Time per eval

<table>
<thead>
<tr>
<th>Eval type</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orig</td>
<td>0.00012487223</td>
</tr>
<tr>
<td>Large Model</td>
<td>0.0015907775</td>
</tr>
<tr>
<td>Med Model</td>
<td>4.9171695e-05</td>
</tr>
<tr>
<td>Small Model</td>
<td>5.14325595e-06</td>
</tr>
</tbody>
</table>
- features determined by radius and number neighbors
- basic regression
- scaling in RVO
- RVO uniform sampling
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

M. L. Jur van den Berg and D. Manocha.  
Reciprocal velocity obstacles for real-time multi-agent collision avoidance.  

M. Kalisiak and M. van de Panne.  
Faster motion planning using learned local viability models.  

J. Miura.  
Support vector path planning.  
A. Y. Ng and S. Russell.
Algorithms for inverse reinforcement learning.

S. J. Russell and A. Zimdars.
Q-decomposition for reinforcement learning agents.

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