Sampling Beats Fixed Estimate Predictors for Cloning Stochastic Behavior in Multiagent Systems

Brian Hrolenok[1] Byron Boots[1] Tucker Hybinette Balch[1]

[1]School of Interactive Computing
Georgia Institute of Technology
{bhroleno,bboots,tucker}@cc.gatech.edu

Thirty-First AAAI Conference on Artificial Intelligence

Problem Statement

Learning Executable Models of Multiagent Behavior

- Generative model for simulation
- Use ML techniques to handle lots of data

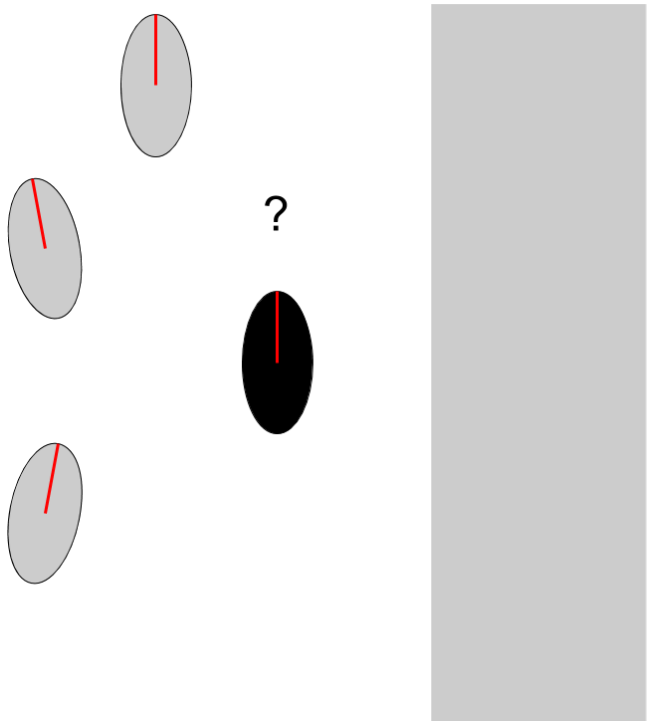
Applications

- Prediction: navigation in crowds
- Modeling: inferring social structures
- Design: swarm search-and-rescue

Focus of this talk: What are the right metrics?

photo credit: Paul Asman and Jill Lenoble

Using observational data to clone behavior



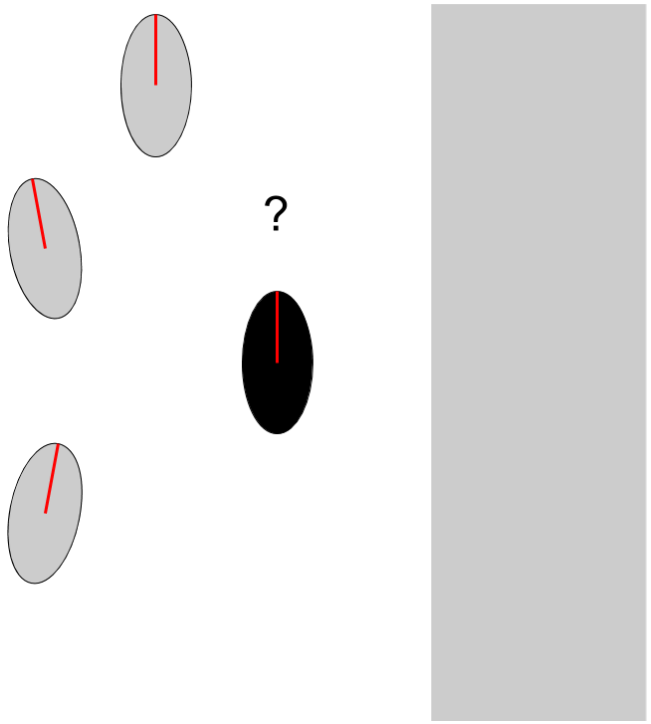
(1) Decide on model class $\hat{f}$ and important features ϕ.

(2) Collect observations of behavior: $D = \{(s_i, b_i)\}$.

(3) Learn $b_i \approx \hat{f}(s_i)$ given D

(4) Given any s_{new}, predict $b_{new} = \hat{f}(s_{new})$.

Using observational data to clone behavior



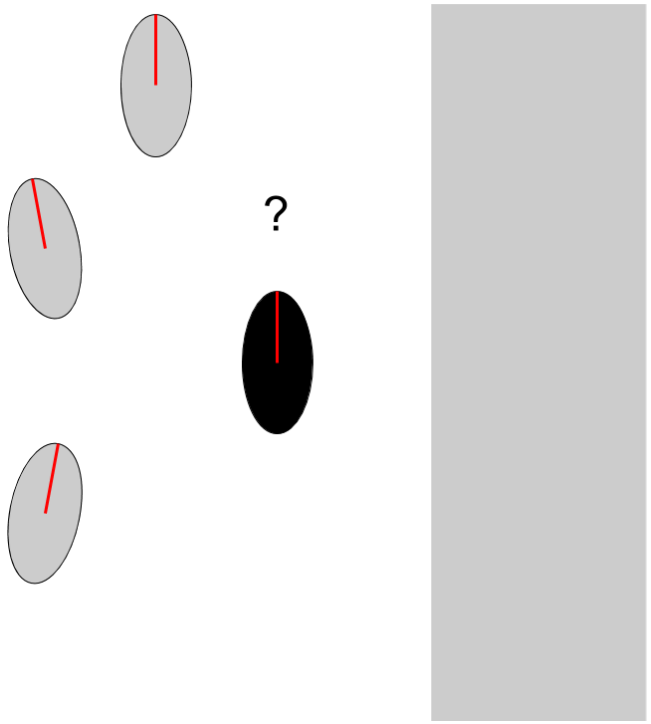
(1) Decide on model class $\hat{f}$ and important features ϕ.

(2) Collect observations of behavior: $D = \{(s_i, b_i)\}$.

(3) Learn $b_i \approx \hat{f}(s_i)$ given D

(4) Given any s_{new}, predict $b_{new} = \hat{f}(s_{new})$.

Using observational data to clone behavior



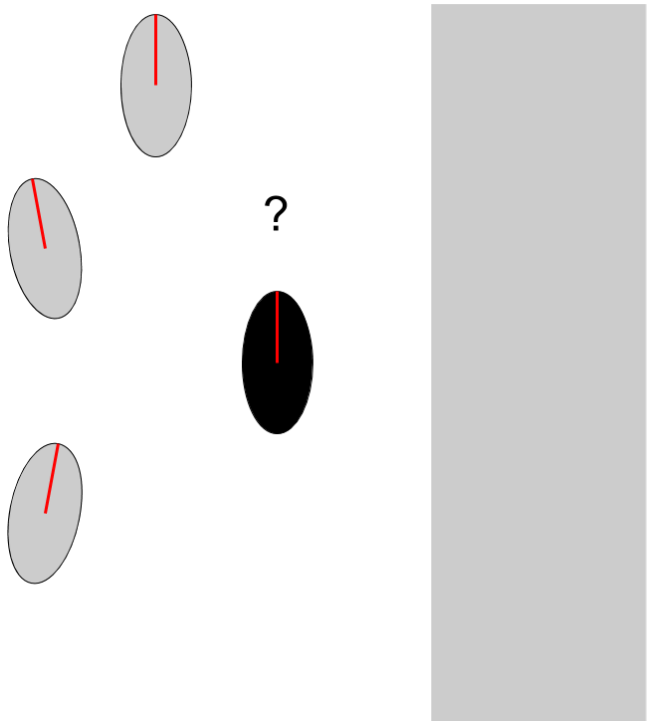
(1) Decide on model class $\hat{f}$ and important features ϕ.

(2) Collect observations of behavior: $D = \{(s_i, b_i)\}$.

(3) Learn $b_i \approx \hat{f}(s_i)$ given D

(4) Given any s_{new}, predict $b_{new} = \hat{f}(s_{new})$.

Using observational data to clone behavior



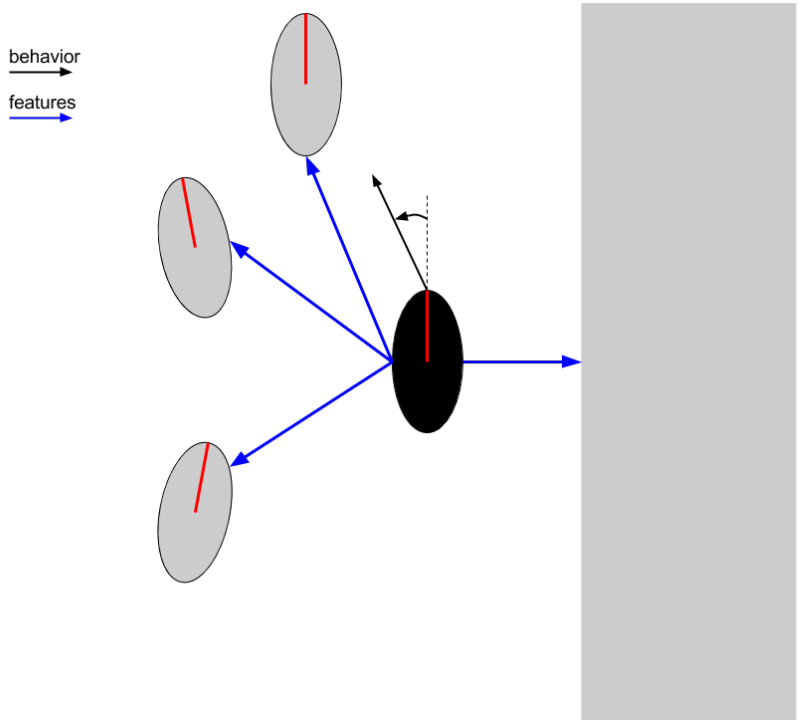
(1) Decide on model class $\hat{f}$ and important features ϕ.

(2) Collect observations of behavior: $D = \{(s_i, b_i)\}$.

(3) Learn $b_i \approx \hat{f}(s_i)$ given D

(4) Given any s_{new}, predict $b_{new} = \hat{f}(s_{new})$.

A simple deterministic model of fish schooling



(1) Linear Model: $\hat{f}(s_i) = \langle \mathbf{W}, \phi(s_i) \rangle$.

(2) Multitarget tracking on video $\rightarrow D$.

(3) Ordinary least-squares: $\mathbf{W} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{b}$.

(4) Simulate: compute agent observation $\phi(s)$, agent action $\hat{f}(s)$, updated state s'. Repeat.

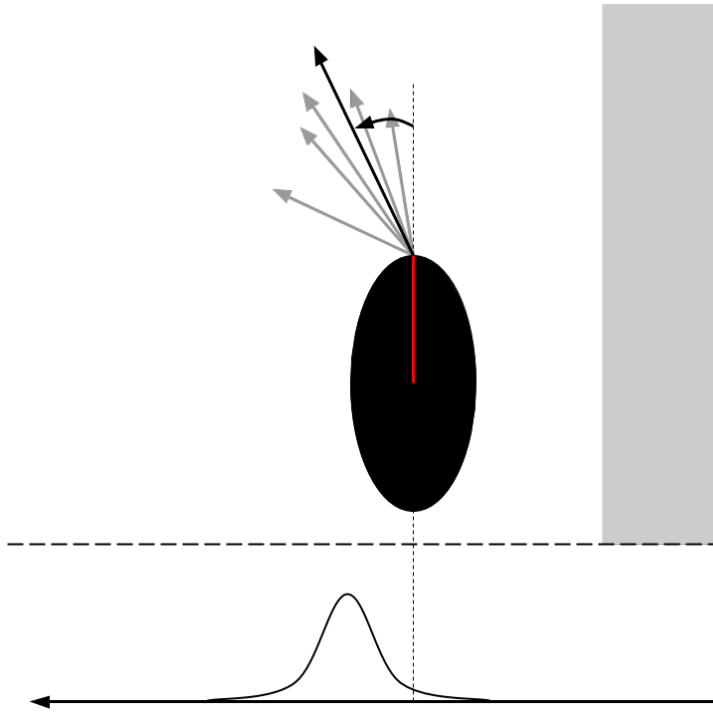
Stochastic behaviors

What if f is not deterministic?

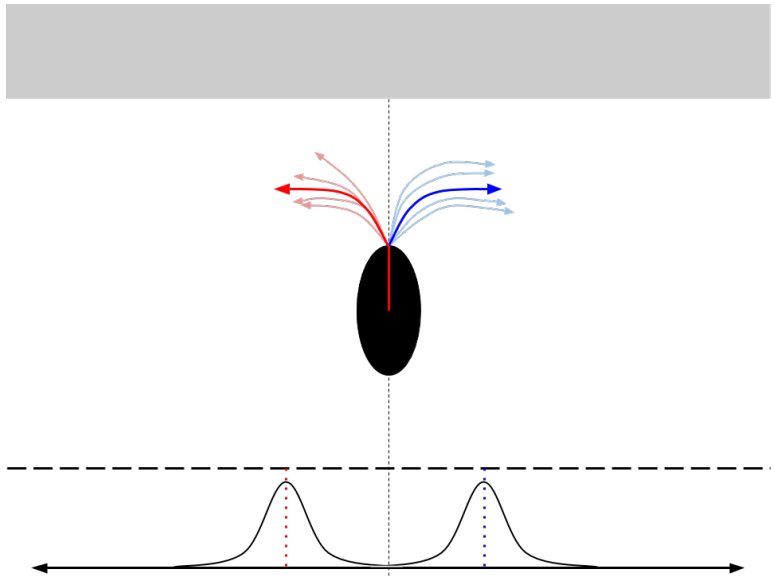
- Additive Noise: $b_i = f(s_i) + \varepsilon_i$
- Assume $\varepsilon_i \sim N(0, \sigma)$
- Minimize loss/risk $\rightarrow$ predict central tendency

What if it's not noise?

- Inherent in behavior
- Recast: $b_i \sim f(s_i) = N(\langle W, \phi(s) \rangle, I\sigma)$
- Strong assumption!



An example stochastic behavior



Realistic behaviors can look like this. What should $\hat{f}$ do?

- Pick the mean?
- Sample under the distribution!

Predicting left or right optimally is still bad (error $\approx 50\%$).
Focus on modeling distribution of behavior (without losing too much on prediction).

Modeling the distribution of behavior

How can we model an unknown density?
Kernel Conditional Density Estimation

$$\hat{f}(b|s) = \frac{\sum K_{h_b}(b, b_i) K_{h_\phi}(\phi(s), \phi(s_i))}{\sum K_{h_\phi}(\phi(s), \phi(s_i))}$$

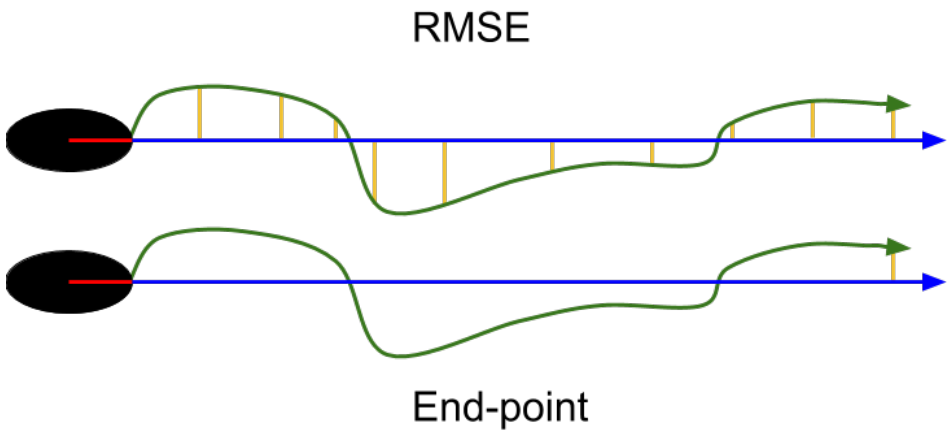
- Simple idea: place "bumps" at all the data points and sum them up
- Kernel (K_{h_b}, K_{h_ϕ}) and bandwidth (h_b, h_ϕ) determine shape

How can we sample from $\hat{f}$ quickly?

- Quick approximation: Use kNN and sample from the nearest neighbors.

Metrics

How should we measure performance? Commonly used metrics:



- RMSE: average error over all points in the test set
- End-point: average difference in final position for all sequences in the test set

These measure predictive performance. Does this capture everything we care about?

Experiments

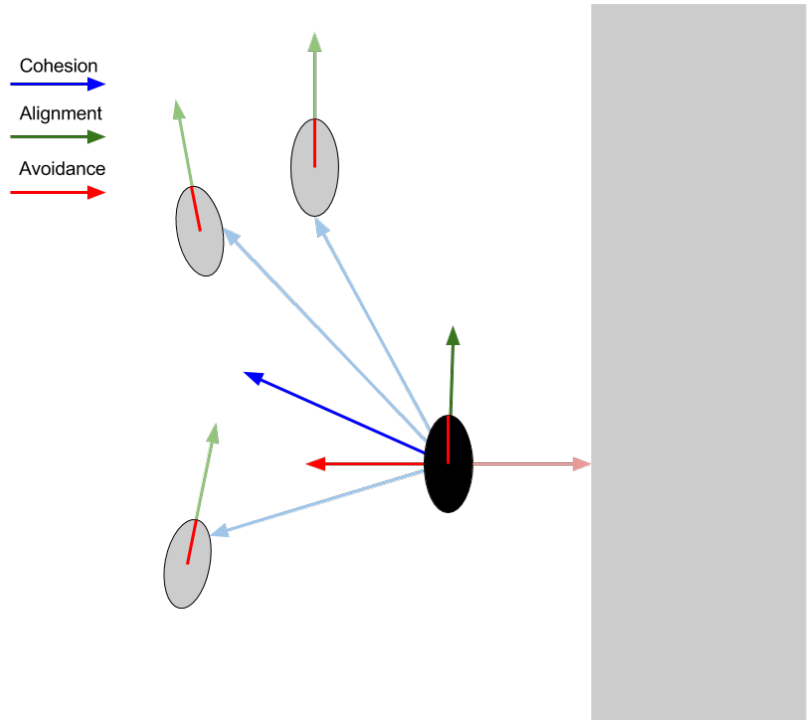
Three experiments

(1) Synthetic deterministic behavior (baseline)

(2) Synthetic stochastic behavior (known behavior function, no noise)

(3) Real fish (unknown behavior function, unknown noise distribution)

Experimental setup



Cloning schooling behavior in fish

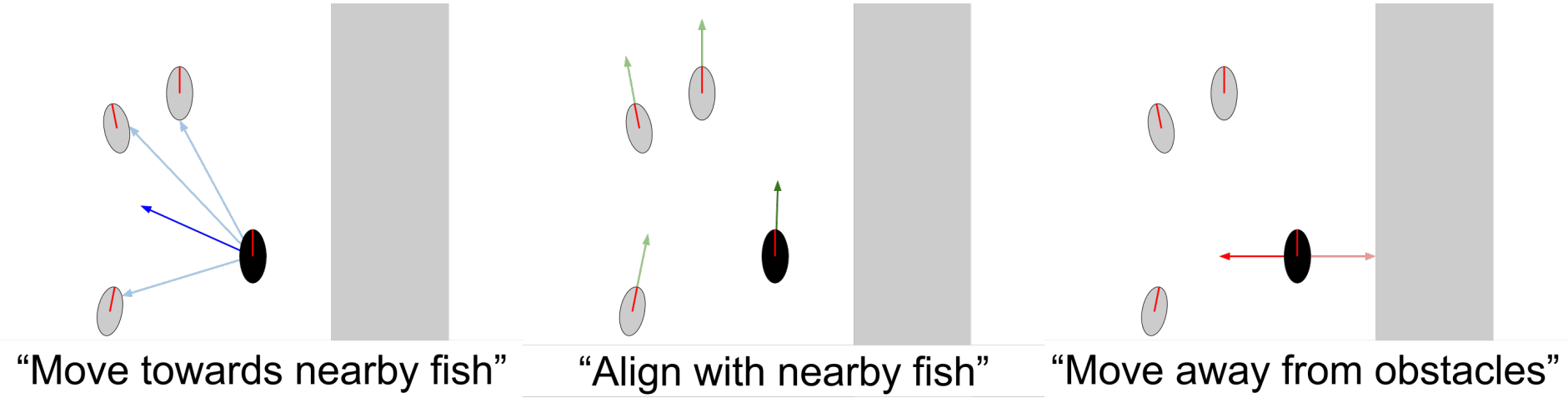
(1) Model class for $\hat{f}$: Lin-Reg, kNN-Reg, kNN-Sample, Features: next slide.

(2) Tracks of fish: $b_i = \left(\frac{dx}{dt}, \frac{dy}{dt}, \frac{d\theta}{dt}\right)$

(3) Train OLS, put data in kNN data structure

(4) Run trained models in simulator

A simple, linear, synthetic behavior

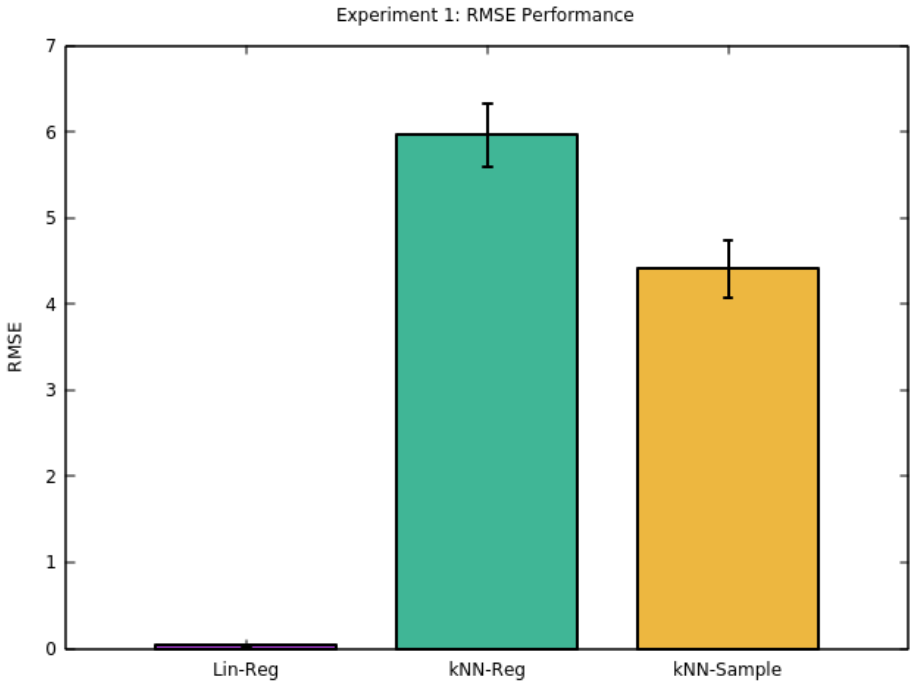


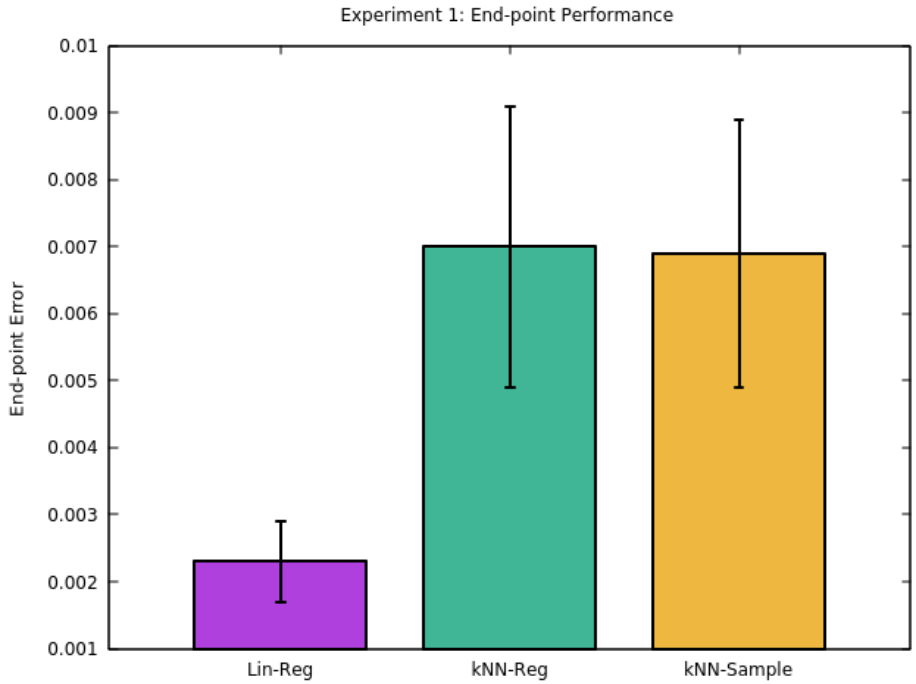
Based on Reynolds' "Boids" model

- Each component is a weighted sum of vectors
- Final behavior is a weighted sum of components
- Model is linear in its features (good candidate for Lin-Reg)
- Produces realistic looking flocking/schooling behavior

Given the individual components as the features ϕ, $\widehat{W}$ found by Lin-Reg closely matches the actual generating model.

Experiment 1: Performance





Predictor	RMSE	end-point
Lin-Reg	$\mathbf{0.0356 \pm 0.0050}$	$\mathbf{0.0023 \pm 0.0006}$
kNN-Reg	5.9675 ± 0.3665	0.0070 ± 0.0021
kNN-Sample	4.4097 ± 0.3328	0.0069 ± 0.0020

- Split data into training/testing on sequence boundaries
- RMSE: error averaged at every point in the test set
- end-point: average difference between final position in test set sequences

Experiment 2: Synthetic stochastic behavior

Same Boids model as before, but this time add a random amount to forward velocity

$$f(x) = \langle W, \phi(x) \rangle + (\varepsilon, 0, 0)$$
$$\varepsilon \sim f_{exp}(x; \lambda) = \lambda e^{-\lambda x}$$

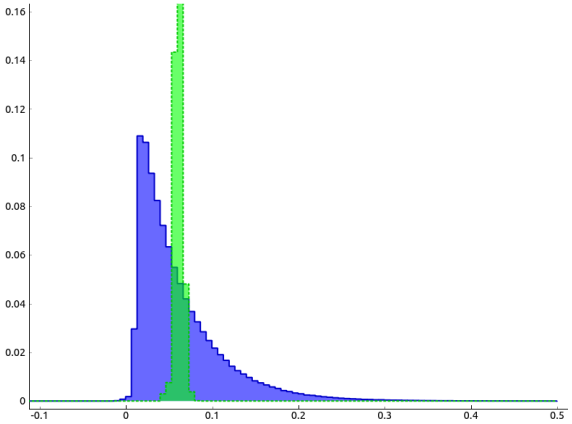
Need a way to measure the similarity of the distribution of $\frac{dx}{dt}$ from $f(s)$ (generating) and $\hat{f}(s)$ (model).

- Qualitative: histograms.
- Quantitative: K-L divergence.

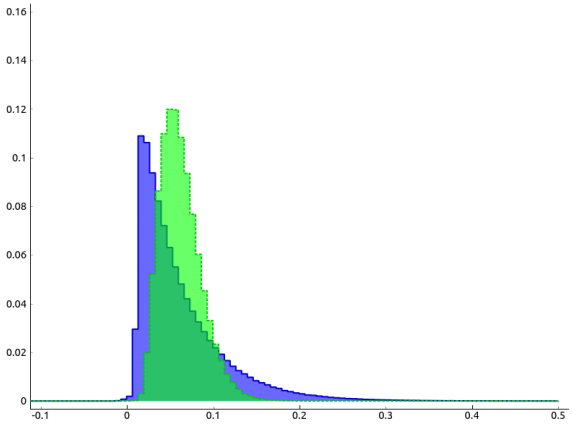
$$D_{KL}(h_f || h_{\hat{f}}) = \sum_i h_f(i) \log \frac{h_f(i)}{h_{\hat{f}}(i)}$$

Experiment 2: Histograms

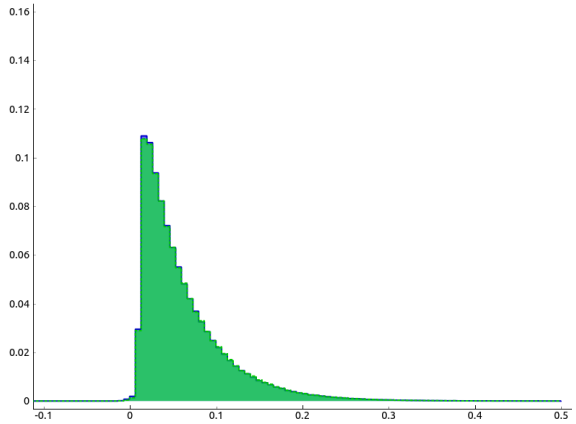
Distributions of $\frac{dx}{dt}$ on the test set



Lin-Reg



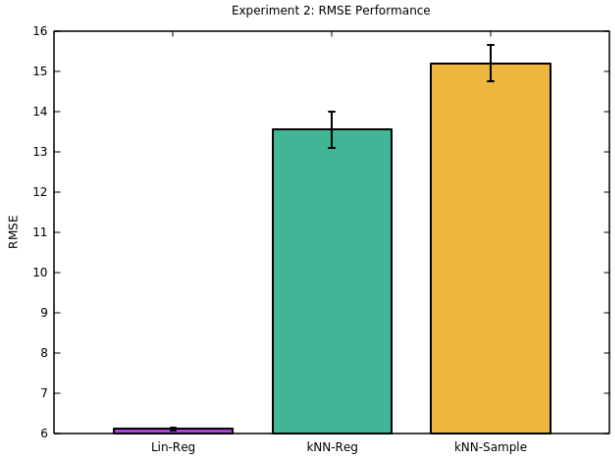
kNN-Reg

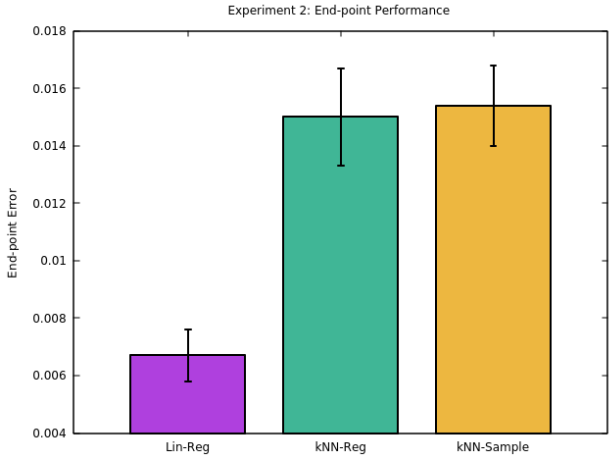


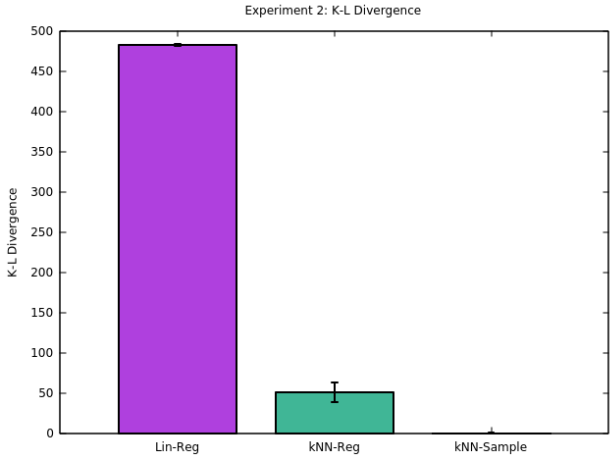
kNN-Sample

- Blue: Generating behavior
- Green: Model behavior

Experiment 2: Performance







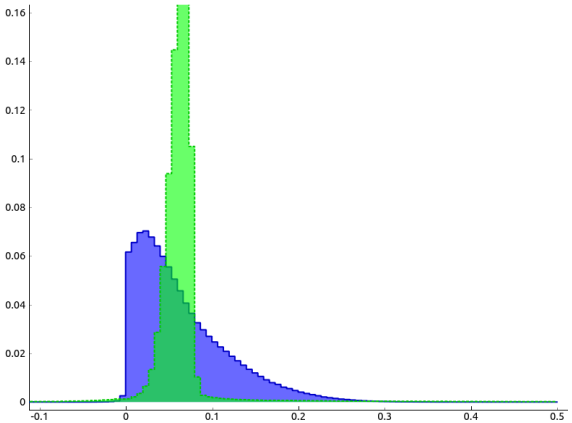
Predictor	RMSE	end-point	K-L divergence
Lin-Reg	$\mathbf{6.1099 \pm 0.0361}$	$\mathbf{0.0067 \pm 0.0009}$	482.6036 ± 0.7310
kNN-Reg	13.5425 ± 0.4549	0.0150 ± 0.0017	51.1153 ± 12.3740
kNN Sample	15.1933 ± 0.4574	0.0154 ± 0.0014	$\mathbf{0.0465 \pm 0.0282}$

kNN-Sample performs worse than Lin-Reg but comparably with kNN-Reg on RMSE and end-point error, but is much better on K-L divergence.

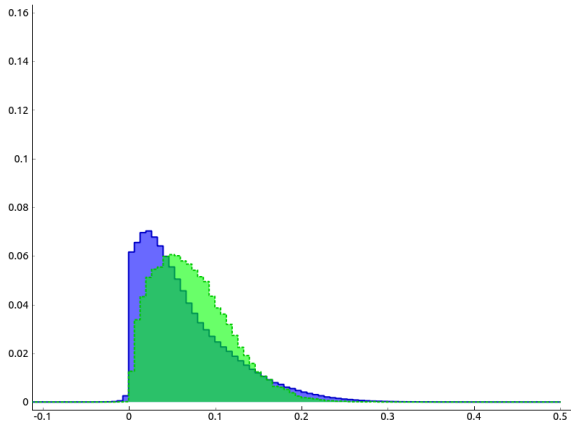
Experiment 3: Real fish

What about real fish?

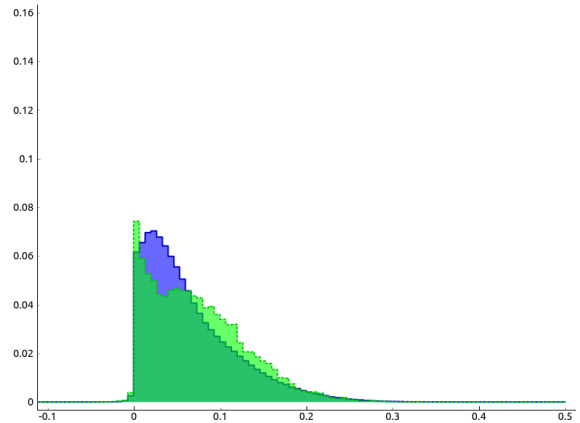
Experiment 3: Histograms



Lin-Reg

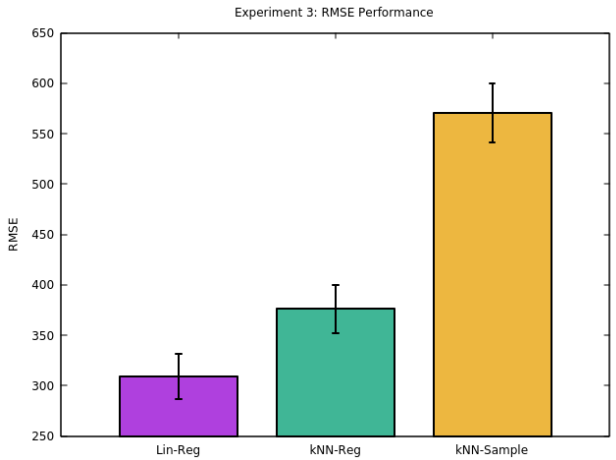


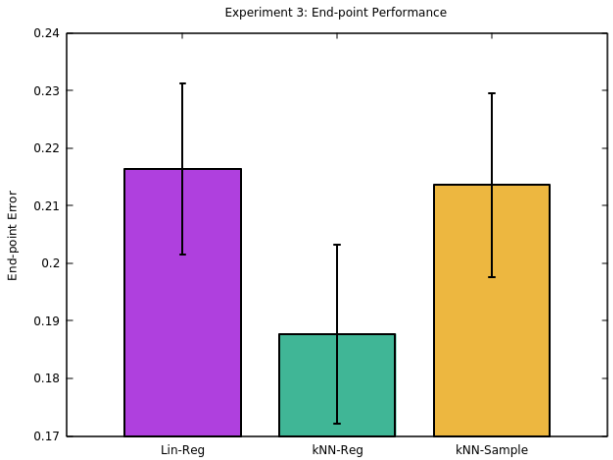
kNN-Reg

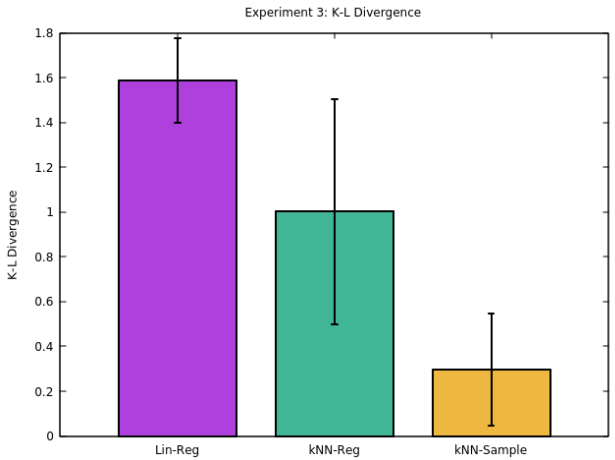


kNN-Sample

Experiment 3: Performance







Predictor	RMSE	end-point	K-L divergence
Lin-Reg	$\mathbf{308.6853 \pm 22.4447}$	0.2164 ± 0.0148	1.5865 ± 0.1894
kNN-Reg	375.9885 ± 24.2010	$\mathbf{0.1877 \pm 0.0156}$	1.0020 ± 0.5021
kNN Sample	570.9620 ± 29.3125	0.2136 ± 0.0160	$\mathbf{0.2957 \pm 0.2497}$

Mixed results. Lin-Reg seems to do best on RMSE, but is outperformed by kNN-Reg on end-point, while kNN-Sample again wins on K-L divergence.

Wrapping up

Conclusions

- For stochastic behaviors, we want to match distributions. Matching behavior in expectation is not enough.
- It's possible to do better at matching distributions without introducing too much predictive error

Future Work

- Fast techniques for optimizing the bandwidths and sampling under the full KCDE (slower than kNN, better approximations)
- Use divergence to construct a loss function directly, then optimize

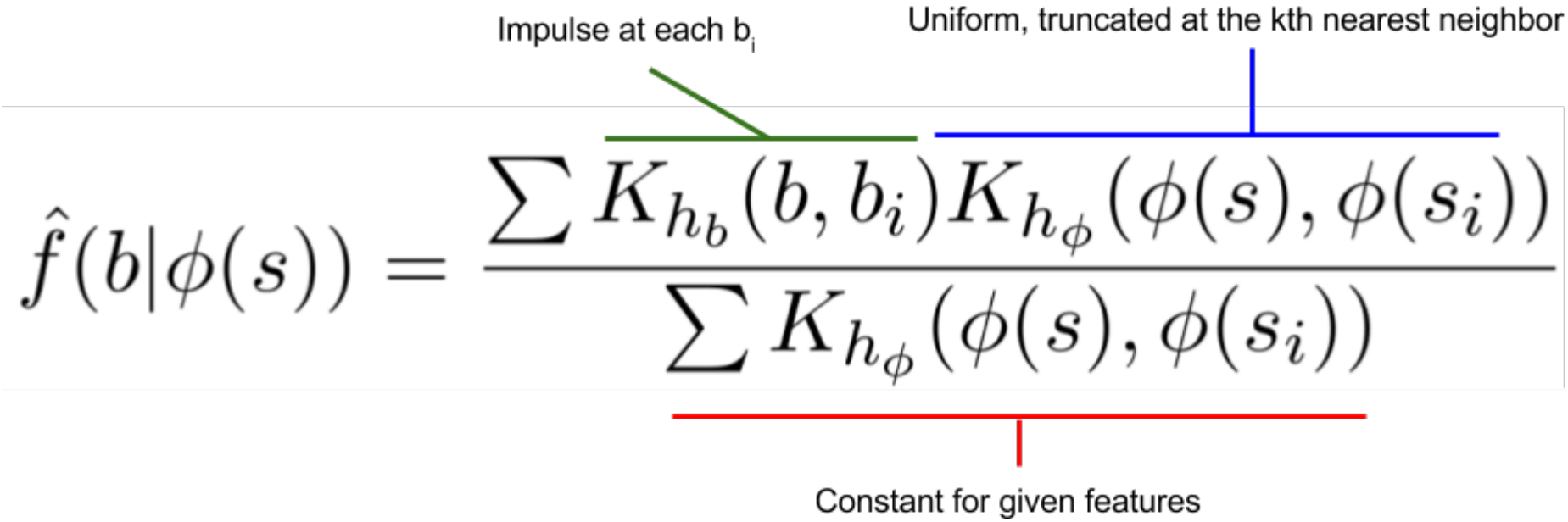
See project page for links to this talk, more images and videos, and code:
`http://www.cc.gatech.edu/~bhroleno/sampling-aaai2017/`

Thanks!

Thank you!

KCDE as KNN

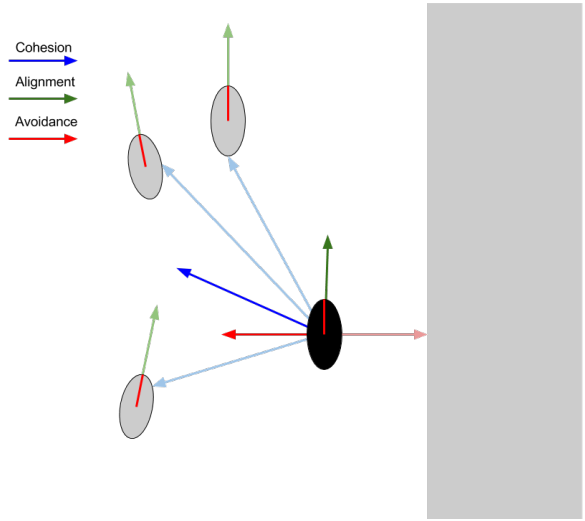
Sampling under $\hat{f}$



- Ignore the denominator, same for all b
- Let $K_{h_b}(\cdot, b_i)$ be impulse at each b_i
- Let $K_{h_\phi}(\cdot, \phi(s_i))$ be uniform truncated at the kth $\phi(s_i)$

Same as picking uniformly at random from among the k nearest neighbors of $\phi(s)$!

Features



$$\phi_i(s) = \frac{1}{n} \sum_{j \neq i} \exp\left\{ \frac{d_j^2(s)}{2\sigma_j^2} \right\} v_{i,j}(s)$$

where $d_j, \sigma_j, v_{i,j}$:

Component	$v_{i,j}$	σ_i
$v_{\text{sep},j}$	Away from agent j $(-(x_j - x))$	Near, 1-2 body lengths (0.1)
$v_{\text{ori},j}$	Heading of agent j (θ_j)	Somewhat near, 2-3 body lengths (0.2)
$v_{\text{coh},j}$	Towards agent j $((x_j - x))$	Majority of school (1.0)
$v_{\text{sep},j}$	Away from obstacle j	Short, within 1 body length (0.05)

Experiment 1: Synthetic deterministic behavior

Boids produces flocking/schooling behavior with a linear combination of simple local rules. Let $f(\phi(s)) = \langle W, \phi(x) \rangle$, with W on the left and $\widehat{W}$ on the right below

Generating	$\dot{x}$	$\dot{y}$	$\dot{\theta}$
ϕ_{sep_x}	-1.0	0.0	0.0
ϕ_{sep_y}	0.0	0.0	-20.0
ϕ_{ori_x}	0.0	0.0	0.0
ϕ_{ori_y}	0.0	0.0	0.1
ϕ_{coh_x}	0.0	0.0	0.0
ϕ_{coh_y}	0.0	0.0	0.8
ϕ_{obs_x}	-1.0	0.0	0.0
ϕ_{obs_y}	0.0	0.0	-40.0
bias	0.0125	0.0	0.0

Recovered	$\dot{x}$	$\dot{y}$	$\dot{\theta}$
ϕ_{sep_x}	-0.9998	0.0	0.0015
ϕ_{sep_y}	-5.1436×10^{-4}	0.0	-19.9995
ϕ_{ori_x}	-1.3071×10^{-5}	0.0	-9.3309×10^{-6}
ϕ_{ori_y}	1.1432×10^{-7}	0.0	0.0998
ϕ_{coh_x}	-3.3333×10^{-6}	0.0	2.8271×10^{-4}
ϕ_{coh_y}	3.1213×10^{-5}	0.0	0.8004
ϕ_{obs_x}	-0.9753	0.0	-0.3991
ϕ_{obs_y}	4.6335×10^{-4}	0.0	-38.7059
bias	0.01250	0.0	-3.5565×10^{-5}

Since Boids is actually a linear model, linear regression can recover the parameters. $||W - \widehat{W}||_F < 0.894$