Data-Driven Models for Zebrafish Motion IDP final

Lukas Krenz
Advisers: Dr. Jacob Davidson (Constance), Nicola Rieke (CAMP)
Supervisor: Prof. Dr. Nassir Navab

June 1, 2018

TUM, Chair for Computer Aided Medical Procedures & Augmented Reality
Collaboration with Couzin Lab (Max Plank Institute for Ornithology/University of Constance)

Introduction

Goal: Develop models for social behavior of juvenile zebrafish that extend to large groups.

Contributions:

- Comparison of models with increasing complexity for modeling interactions between two fish
- Data-driven spatial discretization
- Evaluation of importance of past trajectories
- Development and evaluation of a non-linear recurrent neural network that predicts parameters for mixture of Gaussians

Example **use case**: controlling a fish in a virtual reality environment
"Pilot experiment" for neural network models for collective motion

Zebrafish: Our Input Video

Modeling Fish Motion

Data: Annotated (no tracking needed) videos from 10 experiments with 2 fish swimming, each for 1h. Annotations: positions of both fish and their orientation

Segmentation into c.a. 148000 kicks

Use wall model[1] to ignore areas with high wall influence (any wall closer than c.a. 4.8 cm).

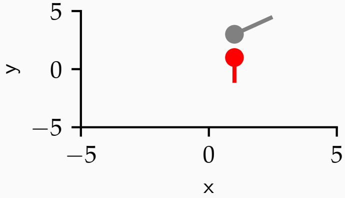
Final data: c.a. 19400 kicks in training set (80% of all kicks)

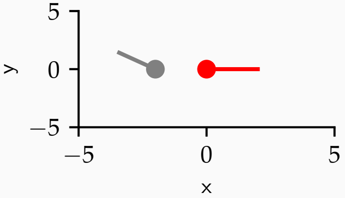
We model kick trajectory (=kick direction and kick length).

---

[1] D. S. Calovi, A. Litchinko, V. Lecheval, U. Lopez, A. P. Escudero, H. Chaté, C. Sire and G. Theraulaz (2018). 'Disentangling and modeling interactions in fish with burst-and-coast swimming reveal distinct alignment and attraction behaviors'. In: *PLoS computational biology* 14.1, e1005933.

Receptive Field[2]—Local coordinate system





Rotate coordinate system s.t. focal fish (red) has angle 0 (parallel to x-axis) and is at origin.

[2] R. Harpaz, G. Tkačik and E. Schneidman (2017). 'Discrete modes of social information processing predict individual behavior of fish in a group'. In: *Proceedings of the National Academy of Sciences*, p. 201703817.

Receptive Field—Discretization

We discretize the area surrounding the fish in the local coordinate system.

We want:

- Symmetry around origin: to distinguish left/right, before/behind
- Bins should have equal number of fish.

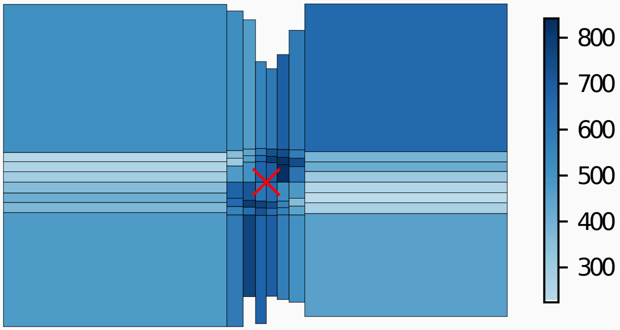
Use 8×8 bins.

Discretizing (with symmetry) axes independently is standard approach.

But: mean number of data points per bin of roughly 2428 ± 8483 (mean $\pm$ std.) for training and 555 ± 1926 for testing.

12 bins completely empty, even for **training set**!

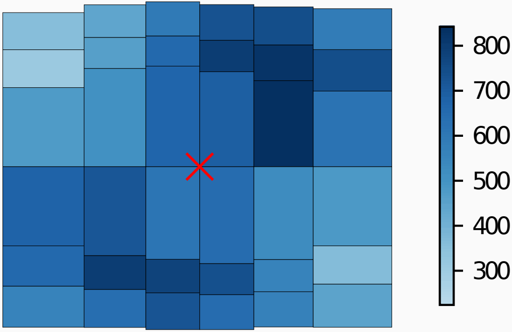
Receptive Field—Data-driven discretization



Mean number of kicks per testing bin

Leads to 2482 ± 212 and 555 ± 162 data points per bin, instead of 2428 ± 8483 and 555 ± 1926.

Receptive Field—Data-driven discretization (zoomed in)



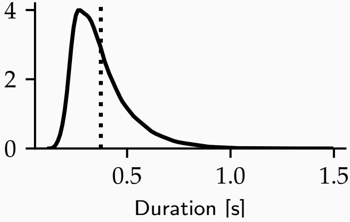
Mean number of kicks per testing bin

Receptive Field—Extracted features

Bin number is one-hot encoded, alternative interpretation as 64 features, each is number of fish currently in bin.

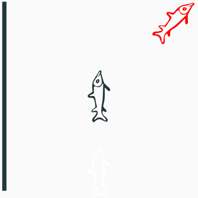
Relative orientation of other fish encoded as unit vector with appropriate orientation, multiplied with bin-feature. Alternative interpretation as mean heading of fish in bin.

Extract this for timesteps 0 s, 0.05 s, . . . , 0.35 s before kick-off-time.



Overall $64 \times (1 + 2) = 192$ features per timestep

Social Models—Linear without memory



No memory: Only current position, etc.

$$\boldsymbol{y^i} = \boldsymbol{X_0^i}\beta_{\mathbf{0}}^{\boldsymbol{i}} + \text{bias} + \varepsilon$$

$$\varepsilon \sim \mathcal{N}\left(0, \sigma^2\right)$$

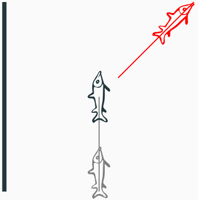
With: $\mathcal{N}(\text{mean}, \text{variance})$ normal distribution

σ standard deviation of residuals

$\boldsymbol{y^i}$ i-th component of output vector

$\boldsymbol{X_0^i}, \beta_{\mathbf{0}}^{\boldsymbol{i}}$ input matrix and weights for i-th component and timestep 0

Social Models—Linear with memory



Memory: Current position and trace

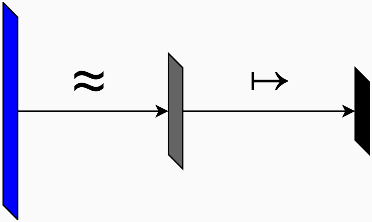
Concatenated: Concatenate features for all timesteps then linear model

Static: Keep spatial weights $\boldsymbol{\beta^i_t}$ static for all t

$$\boldsymbol{y^i} = \sum_t c_t \boldsymbol{X^i_t} \boldsymbol{\beta^i} + \varepsilon$$

with c_t mixing coefficient $c_i \geq 0, \sum_i c_i = 1$, normalized with *softmax*

Social Models—Neural networks



General architecture. Symbols $\approx$ and $\mapsto$ correspond to encoding and decoding. Blue is input, gray hidden and black output.

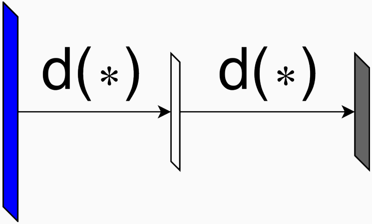
Neural Networks can capture non-linear effects.

Encoder-decoder architecture with

- **Encoder** transforms input features into hidden state
- **Decoder** transforms hidden state into output

We discuss two encoders & two decoders.

Neural Networks—Encoders: Multilayer-Perceptron

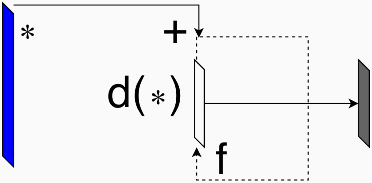


Architecture of MLP encoder. Symbol $d(*)$ corresponds to linear layer, followed by tanh and dropout[3]. Blue is input, gray hidden state.

Multilayer perceptron encoder consists of two stacked layers (Linear -> Tanh -> Dropout) with 64 hidden neurons.

[3] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov (2014). 'Dropout: A simple way to prevent neural networks from overfitting'. In: *The Journal of Machine Learning Research* 15.1, pp. 1929–1958.

Neural Networks—Encoders: Recurrent neural network



Architecture of RNN encoder. Symbol $*$ is linear layer, $d(\,)$ is recurrent dropout[4], $+$ is summation and f to *tanh*. Blue is input, gray hidden state.

Input-to-hidden weights $\boldsymbol{w_{ih}}$, hidden-to-hidden weights $\boldsymbol{w_{hh}}$, biases $\boldsymbol{b}$, recurrent dropout $d(\boldsymbol{x})$ (same mask for all timesteps), learned initial state of size 64 (h_0)

$$\boldsymbol{h_t}(\boldsymbol{X}) = \tanh\left(\boldsymbol{b} + \boldsymbol{w_{ih}}\boldsymbol{X_t} + \boldsymbol{w_{hh}} d(\boldsymbol{h_{i-1}})\right),$$

[4] S. Semeniuta, A. Severyn and E. Barth (2016). 'Recurrent dropout without memory loss'. In: *arXiv preprint arXiv:1603.05118*.

Neural Networks—Decoder: Mixture density networks[5]

Predict mixture of Gaussians

$$p\left(\boldsymbol{y}|\boldsymbol{X}\right) = \sum_{i}^{n} \kappa_i \mathcal{N}\left(\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}\right),$$

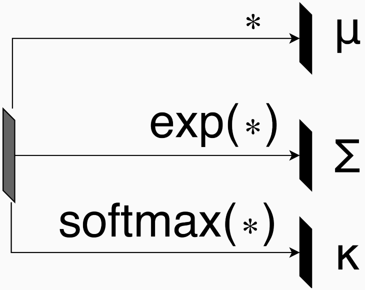
with mixing coefficients κ_i, multivariate normal $\mathcal{N}\left(\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}\right)$ with mean vector $\boldsymbol{\mu_i}$ and covariance matrix $\boldsymbol{\Sigma_i}$

Constraints: $\kappa_i \geq 0$, $\sum_i \kappa_i = 1$, enforced by *softmax*

$\boldsymbol{\Sigma_i}$ valid (diagonal) covariance, diagonals (variances) must be positive, enforced by *exp*.

[5] C. M. Bishop (1994). *Mixture density networks*. Technical Report.

Neural Networks—Decoder: Mixture density networks (cont.)[6]



Architecture of MDN decoder. Symbol $*$ is linear layer. Gray is hidden state and black output.

[6] C. M. Bishop (1994). *Mixture density networks.* Technical Report.

Results—Quantitative

Model	NLL-train	NLL-test	MSE-train	MSE-test
Baseline (Train-Mean)	2.25	2.01	0.558	0.423
MLP-MDN	1.63	**1.60**	0.448	0.373
RNN-MDN	**1.43**	1.69	0.436	**0.371**
MLP-MSE	2.04	1.87	0.452	0.375
RNN-MSE	2.00	1.86	**0.432**	0.377
Linear (w/o time)	2.04	1.87	0.451	0.376
Linear (time conc.)	2.00	1.86	0.432	0.373
Linear (static spatial)	2.04	1.86	0.451	0.374

Results for all models. Baseline: Always predict mean.

Note: Comparison of negative log likelihood NLL is valid, all models can be interpreted as mixture of Gaussians.

Results—Example simulation with RNN-MDN

Conclusion

- Data-driven discretization leads to more equal fish distribution than standard approach.
- Models extend trivially to larger fish groups.
- MDN-models allow sampling, multi-modal distributions and model uncertainty. This is biologically more plausible than our alternatives.
- Non-linear models work but do not show a significant improvement (but should work well for larger groups).