Learning the Hyperparameters of Human Movement

Neil Traft
Department of Computer Science
University of British Columbia
ntraft@gmail.com

Abstract

We address a Gaussian process model selection problem. We wish to model and predict human motion in a crowd. Gaussian processes can be a useful tool for representing the probability distribution over future trajectories of pedestrians in a crowd. However, selecting hyperparameters for the GP can be difficult and prone to overfitting and degeneracy. In addition, it can be difficult to evaluate the performance of different models in order to choose the best one. To bring more robustness to the training, we propose and evaluate multiple novel methods for the optimization of the hyperparameters. To address the problem of evaluation, we employ a version of the statistical similarity metric proposed by Guy, et al. [1]. Ultimately we find that the best method is to train multiple models on individual pedestrians, and use the entropy metric to find the best one.

1 Introduction

Crowd simulation is a hard and interesting problem with many applications. It is already in commercial employ to aid in traffic and transportation engineering, building design, emergency procedures, and much more.

Crowd simulation also has important applications to robotic motion planning. Our primary concern is the development of algorithms for navigation in *dynamic* environments (environments with moving obstacles, mostly pedestrians). Accurate prediction of human motion is crucial for navigation in such environments. A robot must have an accurate estimate of the future state of the obstacles in order to perform robust collision avoidance. Without accurate predictions, the forward simulation of the crowd will result in an "uncertainty explosion" causing the robot to freeze [2].

Accurate predictions of human motion are important for yet another reason: to produce natural, human-like motion on a robotic platform. If it is possible to predict the state of all agents at time $t + 1$, and the robot is one of those agents, then the MAP approximation of the crowd distribution yields the robot's future path. This is an instance of *planning* reducing to *inference*, as described by Trautman et al., 2013 [3].

Trautman and Krause have developed one such method for approximate inference on crowds, and applied it to robot navigation in a crowded cafeteria [3]. The main contribution of this paper is to improve the training of this algorithm.

2 Related Work

2.1 Agent-Based Crowd Simulation

In crowd simulation literature, models fall generally into one of two categories: *macroscopic* and *microscopic* [4]. In this work we are concerned with *microscopic* models, which model each individual agent as a decision-making entity in order to evolve the crowd state.

One of the earliest (and still prominent) models in this category is the Social Force Model [4]. In this model, each agent is modeled as a particle which moves according to attracting and repelling forces determined by its goal and environment information. This is also an example of a model which has many parameters that must be carefully tuned. The model is not learned from data.

There have been some attempts at data driven models in recent years. Lerner et al., 2007 [5] is a prominent example. Here a database of common situations is built from data, and a decision-making agent queries the database for the path they should follow, based on their current situation. The results of this technique were assessed visually, rather than being rated by some objective metric.

In Pellegrini et al. 2009 [6], the authors attempt to learn parameters of a social behavioral model in order to yield improved multi-target tracking. They did not attempt to represent the probability distribution of future agent states, however.

2.2 Robotic Navigation in Crowds

RVO [7] is another method for motion planning in dynamic, crowded environments. It is able to make a theoretical guarantee of collision avoidance, but only under the assumption that all agents are operating under the same family of behavioral rules. Additionally, RVO features many parameters which must be manually tuned or optimized via a search algorithm. In contrast to our method, the parameters of human motion are not learned directly from data, and RVO does not attempt to model the distribution over future trajectories.

Kuderer and Kretzschmar [8] have also approximated the human distribution via Maximum Entropy Inverse Reinforcement Learning [9]. Their method of approximate inference can be seen as a good competitor to ours, though it relies on a mixture of topological variants which may not scale to densely populated environments (see Section VI.F of their RSS 2012 publication [10]).

3 Methods

3.1 Previous Work

3.1.1 Interacting Gaussian Processes

Our starting point in this project is an existing implementation of *Interacting Gaussian Processes* (IGP), a nonparametric stochastic crowd model based on Gaussian processes (GPs). In IGP, the actions of all agents are modeled as a joint distribution:

$$p_{IGP}(\mathbf{f}^{(1)} \dots \mathbf{f}^{(n)} | \mathbf{z}_{1:t})$$

where $\mathbf{f}^{(i)}$ is agent i's trajectory over T timesteps and $\mathbf{z}_{1:t}$ is the set of all observations up to the current time point. The goal for each agent is given as a final "observation" at time T, resulting in the full set of observations $\mathbf{z}_{1:t,T}$. The prior distribution of each agent is represented by a Gaussian process.

To model the joint distribution of all agents, we have an *interaction potential* $\psi(\mathbf{f}^{(i)}, \mathbf{f}^{(j)})$, which substitutes for the conditional $p(\mathbf{f}^{(i)} | \mathbf{f}^{(j)})$ of one agent conditioned on another. The function encodes collision avoidance behavior, such that any set of paths where agents become too close is treated as very unlikely. Thus, the final posterior p_{IGP} is computed by multiplying all the prior probabilities of the agents with their pairwise interactions.

The result is a nonlinear, multimodal distribution, so it can't be evaluated directly. Instead, we sample from the GP priors and resample weighted by ψ. Given this formulation for p_{IGP} and an appropriate weighting w_i for each sample, the ideal paths can now be expressed as the *maximum a-posteriori* (MAP) assignment for the posterior,

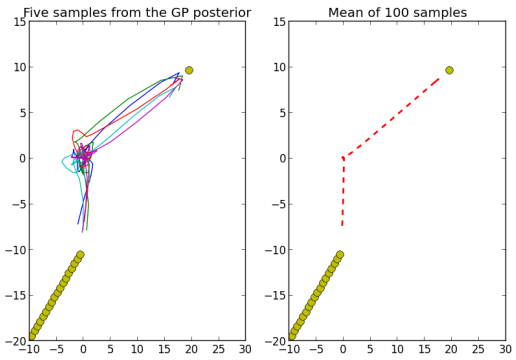
$$(\mathbf{f}^{(1)} \dots \mathbf{f}^{(n)})^* = \arg\max_{\mathbf{f}} \left(\sum_{i=1}^{N} w_i (\mathbf{f}^{(1)} \dots \mathbf{f}^{(n)})_i \right)$$

where $(\mathbf{f}^{(1)} \dots \mathbf{f}^{(n)})_i$ is a set of samples from the Gaussian processes $(\mathbf{f}^{(j)})_i \sim \text{GP}(\mathbf{f}^{(j)}, m_t^{(j)}, k_t^{(j)})$. We take agent i's next position to be $(\mathbf{f}_{t+1}^{(i)})^*$. See [3] for the full details.

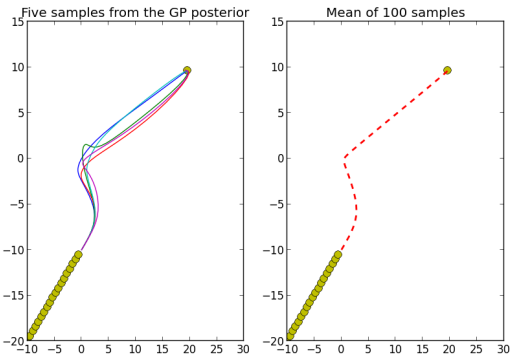
3.1.2 Importance of the Covariance Function

In order to implement IGP, the first and most crucial step is to construct a Gaussian process which fairly well captures the distribution of human movement, as a function of time. To do this we must select both an appropriate kernel and appropriate length-scale parameters for the kernel. This process is illustrated in Figure 1, where we draw sample trajectories from different GPs.

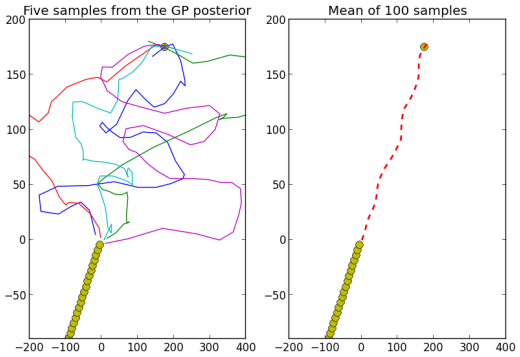
It has been found in our previous work that a good representation of human movement can be built using the summation of linear and Matérn 5/2 kernels, with an additional noise assumption on the training data.



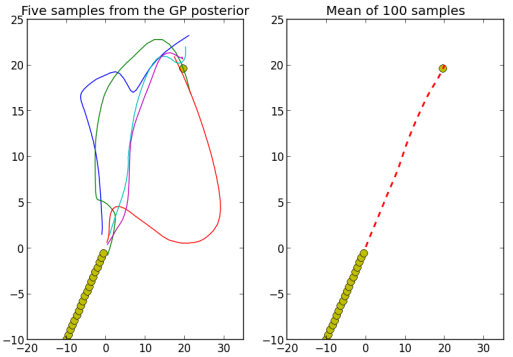
(a) Poor choice of covariance function (squared exponential) and poorly tuned hyperparameters. Notice the attraction toward the mean, $(0, 0)$.



(b) Improved hyperparameters are still not enough to overcome a fundamentally incorrect covariance function.



(c) Good choice of covariance function (Matérn + linear + noise) brings us much closer to the mark, but badly tuned hyperparameters still makes things difficult.



(d) Good covariance function combined with good hyperparameters.

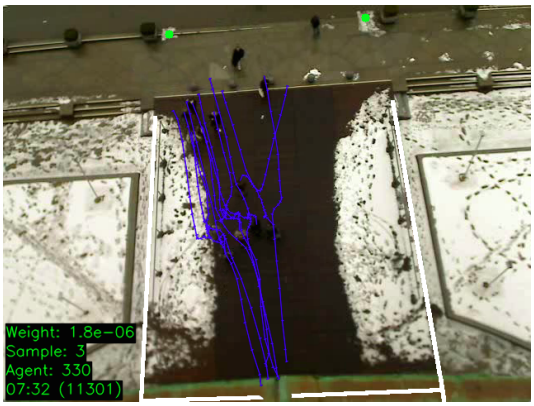
Figure 1: The importance of choosing proper hyperparameters. Hyperparameters must be learned to properly match what we expect from human and robot movement patterns. Hyperparameters must be further tuned for a particular scene, since they are closely related to the scale of our coordinate frame.

3.1.3 Training of the Hyperparameters

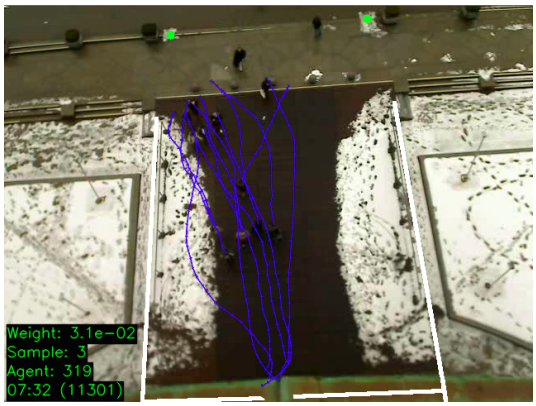
Assuming that we have made a good choice of covariance function, we still need to select the hyperparameters for the function. Normally, we would optimize the likelihood of the training data w.r.t. the hyperparameters, as described in Chapter 5 of *Gaussian Processes for Machine Learning* [11]. The simplest way to do this would be to select an agent whose path is generally "representative" of human motion, and train on them. It is not directly possible to train on multiple agents at once.

This difference arises because our problem has a fundamentally different nature than that of other GP applications. In our case, rather than having a single training corpus, *each pedestrian* constitutes *its own* set of training data for the Gaussian process. There is no single underlying function we wish to model, because every pedestrian forms a new function. Our problem is to find hyperparameters that represent this entire *class* of functions.

This introduces vulnerabilities in the training process where the training procedure may overfit the individual chosen or produce a degenerate covariance due to some kinks or noisiness in the path. See Figure 2 for a qualitative example of what can happen if you make a poor choice of pedestrian.



(a) Poor hyperparameter learning. Agent #194 only.



(b) Better hyperparameter learning. Agents #194-202.

Figure 2: An illustration of how poor choice of exemplar agent can hurt you. If you chose pedestrian #194 to train on, you would get the paths shown in (a). Notice the way they kink up in the middle and shoot out at the ends. Our method can take #194 and a handful of other pedestrians, and learn good hyperparameters from them (b). (Best viewed on a screen. Zoom in for more detail.)

Given these issues, we would like a more reliable way to train the hyperparameters. We would also like, given multiple candidates, a way to select the set of hyperparameters which more closely resembles human movement.

3.2 Contributions

To address the problem of only being able to optimize the hyperparameters with respect to a single pedestrian, we propose three different methods for optimization w.r.t. multiple pedestrians:

- *Joint Optimization*
  1. Concatenate pedestrian paths into a single parametric function. When doing so, shift the paths in time so that they are very far apart from each other on the timeline. This will ensure that their covariance is negligible.
  2. Train as normal on this much larger "path".
- *Bagged Optimization*
  1. Optimize the hyperparameters separately on each agent, as you would normally.
  2. Take the mean of the hyperparameters as the final result.
- *Stochastic Optimization*
  1. Optimize the hyperparameters for a single agent, as normal.
  2. When you move on to the next agent, do *not* use a new set of hyperparameters, but rather use the previous ones as an initial starting guess.
  3. You may choose to apply early stopping to each agent, to prevent overfitting to any one agent.
  4. Continue cycling through the agents in this way until convergence.

In order to determine which one of these methods is best, we should have an objective way to measure the performance of the models they produce. For this we propose a variation on the entropy metric from Guy et al. 2012 [1]. We provide the details of this statistic in the next section.

3.3 Entropy Metric

In our problem, the data are xy-positions on a plane, and so they can be visualized, as we have done earlier in our paper. However, beyond a certain accuracy, the human eye cannot necessarily perceive the best or most "human-like" motion. Furthermore, when it comes to prediction, we can only judge similarity to the true paths, and cannot easily evaluate the probability of that path, or visualize the distribution over an evolving function of time.

We will never have access to the true distribution of possible futures, but we can objectively evaluate our resemblance to a recorded data set. Wolinski et al. 2014 give a brief overview of such crowd prediction metrics [12]. These include:

- Absolute position difference
- Path length
- Inter-pedestrian distance
- Progressive difference (absolute difference when the simulation is re-initialized at each timestep)
- Vorticity (a fluid dynamics measurement)
- Fundamental diagram (a measure of pedestrian flow rate)

Since we are primarily concerned with *microscopic* crowd models, we are less interested in the last two metrics. There are some problems with the other metrics. The behavior of a crowd is highly stochastic, and small changes can quickly compound. Thus these errors may unfairly penalize a simulation that made a bad step early on. We should instead treat any ground truth dataset as a sample from an unknown distribution and perform a statistical analysis on the behavior represented from the example motion.

To this end, Guy et al. proposed a statistical similarity measure for crowd dynamics [1]. Broadly speaking, the entropy metric seeks to measure the size of the prediction error. Given a crowd state $\mathbf{x}_t$, we wish to measure the *variance* of the prediction $\mathbf{x}_{t+1}$. To summarize,

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t) = \hat{f}(\mathbf{x}_t) + \mathbf{m}_t, \quad \mathbf{m}_t \sim \mathcal{M} \tag{1}$$

where f is the true transition function, $\hat{f}$ is the prediction, and $\mathbf{m}_t$ is the prediction error. This means the error is characterized by the distribution $\mathcal{M}$, which is assumed to be Gaussian with 0 mean. Furthermore, the error for each agent is assumed to be independent (i.e. the predictor is not biased toward any particular agent). Thus, the distribution can be fully characterized by the *per-agent variance* M.

Thus, to calculate the entropy metric we must only estimate the expected value $\mathrm{E}[M]$ and compute the entropy,

$$H(\mathcal{M}) \propto H(\mathcal{N}(\mathbf{0}, M)) = \frac{1}{2}\log((2\pi e)^d \det(M)) \tag{2}$$

The difficulty in [1] comes from the fact that they wish to estimate the *latent, true* position and velocity $\mathbf{x}_t = [p_x, p_y, v_x, v_y]$ from only the *observed, noisy* position $\mathbf{z}_t = [p_x, p_y]$. Since the true state $\mathbf{x}_t$ is unknown, estimating the variance M is non-trivial. They accomplish this through an Expectation-Maximization algorithm that estimates the states given the prediction (E-step), and the prediction given the states (M-step), in turn. For full details and algorithm, see their paper.

Since our prediction system deals solely in (x, y) positions, our latent state is the same as our observed state. Furthermore, our dataset is manually annotated, so we can fairly assume that the observation noise is negligibly small. Therefore, in this work we have assumed that we *can* observe the true state ($\mathbf{z}_t = \mathbf{x}_t$), allowing us to do away with the Expectation step of the algorithm. Thus we can simply estimate the variance M using the following algorithm.

Algorithm 1: Expected variance M

$M = \mathbf{0}^{2\times 2}$;
$Z = 0$;
for *each timestep* t **do**
  for *each simulation sample* m **do**
    $\mathbf{x}_{t+1} = \hat{f}(\mathbf{x}_t)$;
    for *each agent* a **do**
      $\text{err} = \mathbf{z}_{t+1}[a] - \mathbf{x}_{t+1}[a]$;
      $M += \text{err}(\text{err})^T$;
      $Z += 1$;
    end
  end
end
$M /= Z$;

4 Analysis

In this section we showcase the usefulness of our entropy metric, as well as use it to report the performance of our proposed hyperparameter optimization methods.

The dataset to be used is the ETH Walking Pedestrians (EWAP) dataset from [6]. It can be obtained from [13]. Twelve sequences were manually selected from this video. Each sequence features multiple agents interacting with another. Most of the sequences were intentionally selected to be difficult to predict without some model of the collision avoidance behavior of individuals (sparsely populated sequences were discarded).

4.1 Validation of Entropy Metric

First it was necessary to validate that the entropy metric performs as we expect it to. We implemented a simple prediction model which takes the ground truth velocity at time t and perturbs it using additive Gaussian noise. This results a path which is perturbed from the ground truth by a given variance. When we run the entropy metric on a sequence at different noise levels, we expect the score to increase logarithmically with the noise variance.

In Figure 3 we see that this logarithmic scale is exactly what arises. This confirms to us that we can reliably use the metric as an error measure in the comparison of different prediction models.

4.2 Hyperparameter Optimization Experiments

We tested four different methods for optimizing the hyperparameters of a human motion model, as outlined in Section 3.2. In all cases, the agents used in training were excluded from the validation set. Only the hyperparameter values were changed in each test; all other parameters were held fixed. The test results on the remaining sequences are summarized in Table 1.

We see that performance generally improves with number of pedestrians included (see Figure 4). We interpret this result to mean that our multiple-pedestrian training methods are able to capture a wider variety of human motions and avoid overfitting and degeneracy.

We also see that among the multiple-agent training methods, the *bagging* method gives the best result according to the entropy metric. We find this at least somewhat surprising as there is not necessarily any statistical validity in the averaging of hyperparameters from different (though similar) datasets.

None of the stochastic methods we tried produced good results, and in most cases they produced degenerate covariance matrices, so we do not include their results here.

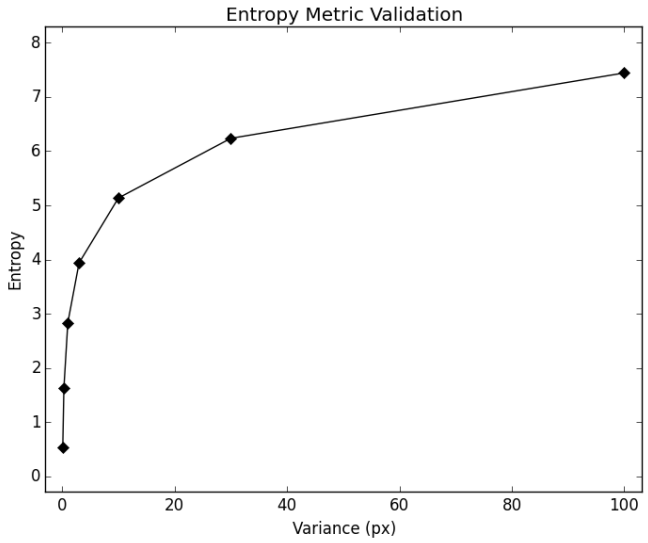


Figure 3: The entropy metric, evaluated on sequence #10 for noise variances in the range [0.1, 100]. This confirms to us that the entropy metric is monotonically increasing with the error in the prediction, making it a valid statistic for performance measurement. We also note the approximate range of values we can expect to see for different values of the per-agent variance M.

Sequence	194 Single	195 Single	198 Single	199 Single	194, 197 Joint	194-202 Joint	194-202 Bagged	194-224 Bagged	174-224 Bagged
1	5.098	4.416	4.82	**4.3**	4.907	4.732	4.671	4.537	4.643
2	3.392	2.364	2.545	**2.195**	2.477	2.392	2.345	2.236	2.286
3	4.65	3.868	4.319	**3.608**	4.337	4.1	3.975	3.908	3.942
4	-	3.741	4.042	**3.519**	3.816	3.709	3.7	3.618	3.697
5	-	3.024	3.457	**2.91**	3.234	3.128	3.142	3.049	3.129
6	-	3.243	3.554	**2.877**	3.55	3.347	3.024	3.028	3.087
7	-	3.487	3.859	**3.319**	3.72	3.535	3.499	3.45	3.526
8	-	3.074	3.074	**2.661**	2.993	2.866	2.722	2.727	2.791
9	-	3.892	4.142	**3.527**	4.157	3.965	3.708	3.664	3.738
10	-	2.763	3.226	**2.515**	3.194	3.084	2.718	2.699	2.756
11	-	2.32	2.726	**2.142**	2.62	2.458	2.321	2.255	2.356
12	-	1.427	1.479	**0.819**	1.533	1.388	1.009	1.035	1.088

Table 1: Hyperparameter Optimization. The header lists the agent(s) used to train the hyperparameters. For the cases with multiple agents, either the *Joint Optimization* or the *Bagged Optimization* are used. Lower scores are better (exponentially better, since the metric is on a log scale). The missing values are because the model produced degenerate covariance matrices for those test cases (remember that agent 194 was our classic example of poor training). The best performer is in **bold** and the second-best is in blue. This shows us that in general, a bagged model is a safer bet, but it can be out-performed with a clever choice of single agent.

4.3 Interaction Algorithm Experiments

Armed now with a reasonable prior for an individual agent's motion, we can experiment with different methods for forming the joint posterior over all agents. We use the entropy metric to evaluate variations on our algorithm to see if they can bring about an improvement. The results are summarized in Table 2.

We test three different methods:

- *Non-interacting Gaussian Processes*: Samples from the agent GPs are *not* weighted by their proximity to other agents. An agent's future path is simply the mean of its samples.

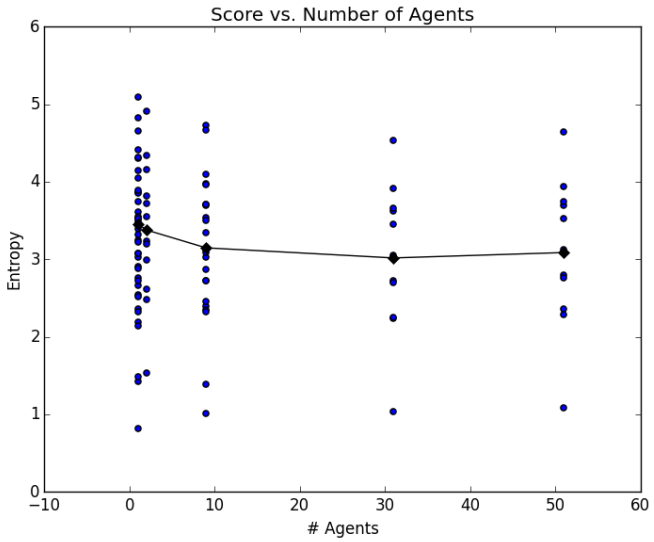


Figure 4: A rough sketch of how performance improves with number of pedestrians included in the optimization (lower is better). However, there may be a limit to this, as it slightly increased when we tried a very large pool of agents.

- *IGP with Weighted Averaging*: Each sample from the joint agent probability distribution is weighted by the Interaction Function ψ. An agent's future path is the weighted mean of samples (the expected value of the posterior).
- *IGP with Weighted Resampling*: Importance sampling is performed on the agent samples and the agent's future path is taken as the MAP statistic (this is the standard IGP algorithm outlined in Section 3.1.1).

The question we ask is, "How much is interaction helping us to determine the direction of motion? Can we do just as well with an approximation to the interaction?"

The surprising answer to this question is that, yes, we can do just as well. Now that we have a reliable method of comparison, we can see clearly that our interaction function is not helping, and may in fact be hurting our prediction. This is the second non-intuitive result made visible to us by our use of the entropy metric.

Sequence	Unweighted	Weighted Mean	Weighted Sampling
1	**4.498**	4.539	4.542
2	**2.231**	2.239	2.250
3	**3.703**	3.875	3.905
4	**3.576**	3.624	3.622
5	**3.031**	3.051	3.053
6	**2.844**	3.025	3.056
7	**3.391**	3.447	3.446
8	**2.637**	2.728	2.723
9	**3.509**	3.686	3.692
10	**2.463**	2.707	2.695
11	**2.161**	2.287	2.301
12	**0.724**	1.006	1.037

Table 2: Interaction Algorithm Tests. Best performance is marked in **bold**. The fact that the non-interacting version of the algorithm won means we've got some work to do, or it may mean that our entropy metric has some weaknesses.

5 Conclusion and Future Work

The crowd entropy metric has allowed us to compare models and determine the best one. We have made the selection of hyperparameters more straightforward and robust by using the metric as a cross-validation tool, combined with novel hyperparameter learning methods.

We have also gained insights into our model, specifically that our interaction function is not modeling the crowd very well. However, this may be due to the simplifying assumptions we made in Section 3.3, which allowed us to leave out the E-step of the original algorithm. It would be prudent to implement the full algorithm and investigate how that affects the results. We would also like to loosen the assumption of a known goal for each agent. This is an unrealistic assumption and the interaction potential may be helpful in situations with less goal information.

One disadvantage of the entropy metric is that it is expensive to compute, and thus not really suitable for an in-depth search of the parameter space of our model. For future work, Wolinski et al. [12] suggest to implement automatic parameter optimization using various simpler metrics for validation, and finally using the (more expensive) entropy metric for reporting the final performance.

References

[1] S. J. Guy, J. van den Berg, W. Liu, R. Lau, M. C. Lin, and D. Manocha, "A statistical similarity measure for aggregate crowd dynamics," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, p. 190, 2012.

[2] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 797–803, Oct. 2010.

[3] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: the case for cooperation," *2013 IEEE International Conference on Robotics and Automation*, pp. 2153–2160, May 2013.

[4] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[5] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," in *Computer Graphics Forum*, vol. 26, pp. 655–664, Wiley Online Library, 2007.

[6] S. Pellegrini, a. Ess, K. Schindler, and L. van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," *2009 IEEE 12th International Conference on Computer Vision*, pp. 261–268, Sept. 2009.

[7] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*, pp. 3–19, Springer, 2011.

[8] H. Kretzschmar, M. Kuderer, and W. Burgard, "Learning to predict trajectories of cooperatively navigating agents,"

[9] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning.," in *AAAI*, pp. 1433–1438, 2008.

[10] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard, "Feature-based prediction of trajectories for socially compliant navigation.," in *Robotics: Science and Systems*, 2012.

[11] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[12] D. Wolinski, S. J Guy, A.-H. Olivier, M. Lin, D. Manocha, and J. Pettré, "Parameter estimation and comparative evaluation of crowd simulations," in *Computer Graphics Forum*, vol. 33, pp. 303–312, Wiley Online Library, 2014.

[13] "Ethz - computer vision lab: Datasets." `http://www.vision.ee.ethz.ch/datasets/index.en.html`. Accessed: 2014-03-01.