Noise Contrastive Meta-Learning for Conditional Density Estimation using Kernel Mean Embeddings

Jean-François Ton University of Oxford
Lucian Chan University of Oxford
Yee Whye Teh University of Oxford
Dino Sejdinovic University of Oxford

Abstract

Current meta-learning approaches focus on learning functional representations of relationships between variables, *i.e.* estimating conditional expectations in regression. In many applications, however, the conditional distributions cannot be meaningfully summarized solely by expectation (due to *e.g.* multimodality). We introduce a novel technique for meta-learning conditional densities, which combines neural representation and noise contrastive estimation together with established literature in conditional mean embeddings into reproducing kernel Hilbert spaces. The method is validated on synthetic and real-world data, demonstrating the utility of sharing learned representations across multiple conditional density estimation tasks.

1 Introduction

The estimation of conditional densities $p(y|x)$ based on paired samples $\{(x_i, y_i)\}_{i=1}^n$ is a ubiquitous task in the modelling of relationships between x and y. While the problem of regression focuses on estimating the conditional expectations $\mathbb{E}[y|x]$ of responses y given the features x, many scenarios require a more expressive representation of the relationship between x and y. In particular, the distribution of y given x may exhibit multimodality or heteroscedasticity, thus requiring a flexible nonparametric model of the full conditional density. Estimating conditional densities becomes challenging when x and y are multivariate and the sample size is small. Hence, we tackle this problem from a meta-learning perspective, where we utilise the shared learned representation of both response y and features x across different tasks, and estimate the conditional density of these tasks.

Contribution: We develop a new approach for modelling functional relationship, which parallels to the (conditional) Neural Processes [1] [2], and is applicable to a broader set of relationships between random objects, *i.e.* the response y that cannot be represented by a single function $f(x)$ with the features x, for instance multimodality in the y's. We use the framework of conditional mean embeddings (CME) of distributions into reproducing kernel Hilbert spaces (RKHSs) [3, 4], which has a unique representation of a probability distribution.

In this work, we use neural networks to *learn* the feature maps ϕ_x and ϕ_y under the meta-learning framework, *i.e.* consider a number of (similar) conditional density estimation tasks simultaneously. While CME estimation for fixed feature maps is well understood [3, 4], we are concerned with the transition from the CME estimates to the conditional density estimation (CDE) task, while simultaneously learning the feature maps defining CME. We use a noise contrastive estimation (NCE)[5] technique to address this problem, treating CMEs as features in the binary classifier and discriminate between the true and artificially generated samples of (x_i, y_i) pairs.

The proposed method is validated on synthetic and real-world data, including Ramachandran plots extracted from the database [6], and the NYC taxi data [7] to model drop-off locations.

Related work: NCE methods [8, 9, 10] have been used in representations learning, and the work in [8, 9] focuses on learning discrete distributions in the context of Natural Language Processing

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

(NLP). Several studies [11, 12] used RKHSs for density estimation. They considered training kernel exponential family models, with a bottleneck in computing the normalizing constant. In addition, different CME sampling methods have been developed, such as kernel herding [13][14]. Note that the herding methods are used to produce representative samples using CME as reference, while we use the CME as a feature to model the conditional density (more details see Appendix).

2 Background

Throughout this paper we denote the observed dataset by $\mathcal{D} = \{(x_j, y_j)\}_{j=1}^n$, with $x_j \in \mathcal{X}$ and $y_j \in \mathcal{Y}$. We define the learned RKHS/feature maps of inputs X and responses Y as $\mathcal{H}_X/\phi_x$ and $\mathcal{H}_Y/\phi_y$ respectively.

Kernel mean embeddings of distributions provide a powerful framework for representing probability distributions [3, 4]. Formally, given sets $\mathcal{X}$ and $\mathcal{Y}$, with a distribution P over the random variables (X, Y) taking values in $\mathcal{X} \times \mathcal{Y}$, the conditional mean embedding (CME) of the conditional distribution of $Y|X = x$, assumed to have density $p(y|x)$ defined as:

$$\mu_{Y|X=x} := \mathbb{E}_{Y|X=x}[\phi_y(Y)] = \int_{\mathcal{Y}} \phi_y(y) p(y|x) dy. \tag{1}$$

Here $\phi_y(y)$ is a function in the RKHS and can be equivalently written as $k_y(y, .)$. Hence, we can obtain an element $\mu_{Y|X=x}$ of $\mathcal{H}_Y$ for each value of the conditioning variable x. Following [3], the conditional mean embedding can be associated with the operator $\mathcal{C}_{Y|X} : \mathcal{H}_X \to \mathcal{H}_Y$, which satisfies $\mu_{Y|X=x} = \mathcal{C}_{Y|X}\phi_x(x)$, where $\mathcal{C}_{Y|X} := \mathcal{C}_{YX}\mathcal{C}_{XX}^{-1}$ s.t. $\mathcal{C}_{YX} := \mathbb{E}_{Y,X}[\phi_y(Y) \otimes \phi_x(X)]$ and $\mathcal{C}_{XX} := \mathbb{E}_{X,X}[\phi_x(X) \otimes \phi_x(X)]$. As a result, the finite sample estimator of $\mathcal{C}_{Y|X}$ based on dataset $\{(x_i, y_i)\}_{i=1}^n$ can be written as $\widehat{\mathcal{C}}_{Y|X} = \Phi_y(K + \lambda I)^{-1}\Phi_x^T$, where $\Phi_y := (\phi_y(y_1), \ldots, \phi_y(y_n))$ and $\Phi_x := (\phi_x(x_1), \ldots, \phi_x(x_n))$ are the feature matrices, $K := \Phi_x\Phi_x^T$ is the kernel matrix with entries $K_{i,j} = k_x(x_i, x_j) := \langle \phi_x(x_i), \phi_x(x_j) \rangle$, and $\lambda > 0$ is a regularization parameter.

Note that using the finite sample estimator, the CME operator is a solution to a vector-valued ridge regression problem (regressing $\phi_y(y)$ to $\phi_x(x)$), allowing computation to scale linearly in the number n of observations using the Woodbury matrix identity.

Noise contrastive estimation (NCE): NCE [5] converts density estimation into binary classification by learning to discriminate between the noisy artificial/fake data and the real data. Following [5], we set up the experiments in such a way that we see κ times more fake examples than real ones. In our case, the true samples come from the conditional $p(y|x)$, *i.e.* $\{y_i\}_{i=1}^n$, and the fake/noisy ones from the density $p_f(y)$, *i.e.* $\{y_i^f\}_{i=1}^{n\kappa}$. For a given x, assuming that the classifier observes samples from the mixture $\frac{1}{\kappa+1}p(y|x) + \frac{\kappa}{\kappa+1}p_f(y)$, the probability that y arises from the true conditional distribution $p(y|x)$ as opposed to the fake density $p_f(y)$ is given by:

$$P(\text{True}|y, x) = \frac{p(y|x)}{p(y|x) + \kappa p_f(y)} \quad \Longleftrightarrow \quad p(y|x) = \frac{\kappa p_f(y) P(\text{True}|y, x)}{1 - P(\text{True}|y, x)}. \tag{2}$$

Under the assumption that the learned probabilistic classifier attains Bayes optimality, we can deduce the point-wise evaluations of the true conditional density $p(y|x)$ directly from expression (2). We would like to fit $p_\theta(y|x)$ to $p(y|x)$, hence we consider the following parametric density model:

$$p_\theta(y|x) = \frac{\exp(s_\theta(x, y))}{\int \exp(s_\theta(x, y'))dy'} = \exp(s_\theta(x, y) + b_\theta(x)) \tag{3}$$

for some *scoring function* $s_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, following the terminology in [8]. Under this model, the probability that y arises from the true conditional distribution $p_\theta(y|x)$ is given by:

$$P_\theta(\text{True}|y, x) = \frac{\exp(s_\theta(x, y) + b_\theta(x))}{\exp(s_\theta(x, y) + b_\theta(x)) + \kappa p_f(y)} = \sigma\left(s_\theta(x, y) + b_\theta(x) - \log(\kappa p_f(y))\right). \tag{4}$$

Eq.(4) gives us the probabilistic classifier we will adopt, where we will need to construct the scoring function $s_\theta(x, y)$ appropriately, in particular, how $s_\theta(x, y)$ relates to the feature maps ϕ_x and ϕ_y.

3 Methodology

Frist, we map x_i and y_i using feature maps $\phi_x : \mathcal{X} \to \mathcal{H}_X$ and $\phi_y : \mathcal{Y} \to \mathcal{H}_Y$. Then we learned and parametrized the feature maps with neural networks, and both sets of parameters collated into θ. Here, we use finite-dimensional feature maps, but other choices are possible. Next, we compute the Conditional Mean Embedding Operator (CMEO) $\widehat{\mathcal{C}}_{Y|X} : \mathcal{H}_X \to \mathcal{H}_Y$ (see the above section). Given $\widehat{\mathcal{C}}_{Y|X}$, we estimate the conditional mean embedding for any new x^* using

$$\widehat{\mu}_{Y|X=x^*} = \widehat{\mathcal{C}}_{Y|X}\phi_x(x^*). \tag{5}$$

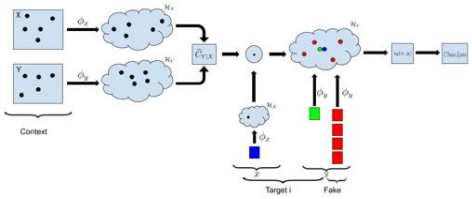

Figure 1: The context data is first passed through the feature maps to construct the CMEO ($\widehat{\mathcal{C}}_{Y|X}$), which is then used to project the new target x (blue) to $\mathcal{H}_Y$, where it is compared to the *True*(green) and *Fake* y's(red)

Note that $\widehat{\mu}_{Y|X=x^*} \in \mathcal{H}_Y$. We then compute $\langle \phi_y(y^*), \widehat{\mu}_{Y|X=x^*} \rangle_{\mathcal{H}_Y} = \widehat{\mu}_{Y|X=x^*}(y^*)$ for any new $y^* \in \mathcal{Y}$, *i.e.* evaluation of the CME at any given new response. This can be interpreted as the "similarity" between $\phi_y(y^*)$ and the prediction $\widehat{\mu}_{Y|X=x^*}$ in $\mathcal{H}_Y$. We expect high similarity when y^* is drawn from the true conditional distribution $Y|X = x^*$, and low when y^* is drawn from the fake distribution, where the true conditional density $p(y|x^*)$ is low. Hence we use the scoring function:

$$s_\theta(x^*, y^*) = \langle \phi_y(y^*), \widehat{\mu}_{Y|X=x^*} \rangle_{\mathcal{H}_Y}. \tag{6}$$

Given a set of true examples $\{(x_j, y_j)\}_{j=1}^n$ and the fake responses $\{y_{i,j}^f\}_{i=1}^\kappa$ associated to each input x_j, where $y_{i,j}^f$ is the i^{th} sample from the fake distribution for data point y_j, we can now train the classifier using model (4) by minimizing the logistic loss

$$\min_\theta \sum_{j=1}^n \left\{ \log\left(1 + \frac{\kappa p_f(y_j)}{\exp(s_\theta(x_j, y_j) + b_\theta(x_j))}\right) + \sum_{i=1}^\kappa \log\left(1 + \frac{\exp(s_\theta(x_j, y_{i,j}^f) + b_\theta(x_j))}{\kappa p_f(y_{i,j}^f)}\right) \right\}. \tag{7}$$

The conditional density estimates can be obtained from (3) with the learned classifier. The details of the fake density and the learning objectives are discussed in the Appendix.

Next, we train our developed model in the meta-learning setting. We consider the case with limited sample for each density estimation task. We adopt the meta-learning framework in order to make use of similar tasks to infer the density. To this end, let $\mathcal{T} = \{T_1, \ldots, T_l\}$ be the set of l conditional density estimation tasks, such that T_q corresponds to the dataset $\mathcal{D}^q = \{(x_i^q, y_i^q)_{i=1}^{m_q}\}$, where $x_i^q \in \mathcal{X}$ and $y_i^q \in \mathcal{Y}$ share the same domains across the tasks. We use an approach similar to the Neural Process (NP) [1], where we define a context set (used to embed the task) and a target set (used to compute the loss and update the parameters of the model) during training. For example, in task q we use m_{cq} samples as context and the remaining $m_{tq} = m_q - m_{cq}$ as target. We use the context set to estimate the CMEO, $\widehat{\mathcal{C}}_{Y|X}$, and evaluate the conditional mean embeddings on the target set using (5).

For each target example, we sample κ fake samples from $p_f(y)$ and represent them in $\mathcal{H}_Y$ using the feature map ϕ_y, so that (4) can be computed for each of these $\kappa + 1$ samples (1 true and κ fakes). We then train the classifier with the labels, *i.e.* True(from data)/Fake(from noise distribution), and learn the parameters θ of neural networks ϕ_x, ϕ_y and b_θ using the objective (7) jointly over all tasks. The resulting feature maps, ϕ_x and ϕ_y, generalize across tasks and can be applied to new, previously unseen datasets. This is done by computing the scoring function $s_\theta(x, y)$ using the CMEO estimated on this new dataset, and insertion into (3). Figure 1 outlines one meta-training step for a given task and illustrates how the loss is computed. We repeat this step for every training task.

4 Experiments

For both synthetic and real-world examples, we compare our method with different conditional density estimations methods, including ϵ-KDE, DDE [11], KCEF [12], and LSCDE [15]. Throughout our experiments we fix $\kappa = 10$ as suggested in [5]. In addition, we include meta-learning algorithms such as Neural Process[1] and a pure neural network version of our framework (MetaNN). in MetaNN, we compute a feature representation of the concatenated (x,y) pairs using a NN, and trained it like our

proposed model, in order to investigate the importance of the CMEO (see the Appendix for a more detailed study). At testing time, we evaluate the method on 100 new tasks with n_c context/training points each. We pass the new context points to our model, and evaluate the density with a simple forward pass in (3). The non meta-learning baselines are trained separately on each of the 100 datasets. We then report the mean log-likelihood of the 100 datasets. High variance in some methods indicated the difficulty of tasks. Hence we use p-values of the one-sided signed Wilcoxon test to confirm that the likelihood of our method, MetaCDE, is significantly higher than all other methods.

Synthetic data: In order to measure the ability of the method in learning the multimodality and heteroscedasticity in the response variable, we construct the datasets as follows: we sample $y_i \sim \text{Uniform}(0, 1)$ and set $x_i = \cos(ay_i + b) + \epsilon_i$, where a and b vary between tasks, with noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. Here, the x can be written as a simple function of y with noise, but not vice versa on the whole range of x, leading to the multimodality of $p(y|x)$. In meta-learning, we use 50 context points and 80 target points for each task. We included experiments with 15 and 30 context points to illustrate the robustness of the methods with varying context data sizes (see Appendix). We also included details of the NN architectures and experimental setup (see the Appendix).

Real-World data: Finding all energetically favourable conformations for flexible molecule in both bound and unbound state is one of the biggest challenge in computational chemistry [16] as the number of possibilities increases exponentially with the dimension. Knowledge about the distributions of dihedral angles in molecules (represented by *Ramachandran plots* [17]) is used in different sampling schemes and it is currently limited by the library curated by chemists. Here, we apply MetaCDE to learn richer relationships between dihedral angles, which can improve the molecule conformation sampling scheme. In our experiement, we consider the cases with 80 data points per task during training and 20 at testing time. In our meta-learning setup, we take 20 context and 60 target points. We have also performed experiments on the NYC taxi dataset, which, due to space constraints, we have added to the Appendix.

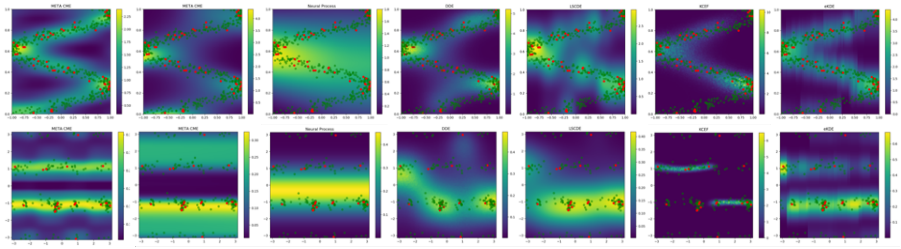

Figure 2: In order ((Top) synthetic- (Bottom) Real- dataset): MetaCDE(ours), MetaNN (ours), DDE, LSCDE, KCEF, ϵ-KDE. The red dots are context points and the green dots are evaluation points.

	MetaCDE	MetaNN	NP	DDE	LSCDE	KCEF	ϵ-KDE
Sythn. Data Mean over 100 log-likelihoods	**197.84 ± 22.45**	172.76±93.56	-81.11±18.53	162.98 ± 69.01	44.95 ± 74.36	-388.30 ± 703.17	116.31 ± 236.99
Sythn. Data P-value for Wilcoxon test	NA	0.0439	< 2.2e-16	8.144e-07	<2.2e-16	< 2.2e-16	2.384e-07
Real. Data Mean over 100 log-likelihoods	**-305.49 ± 46.99**	-317.91±51.36	-426.75±47.31	-399.68 ± 41.30	-407.32 ± 80.19	-724.40 ± 891.64	-485.10 ± 303.49
Real. Data P-value for Wilcoxon test	NA	0.001	< 2.2e-16	1.658e-15	2.579e-14	9.72e-14	2.949e-14

Table 1: Average held out log-likelihood on 100 different synthetic cos tasks. We also compute the p-value for the one sided signed Wilcoxon test with respect to MetaCDE

5 Conclusions and Future Work

In this paper we have introduced a novel method for conditional density estimation in a meta-learning setting. We applied our method to a variety of synthetic and real-world data, with a strong performance in computational chemistry task and an example with NYC taxi data. Owing to the meta-learning framework, experiments indicate that the developed method is able to capture correct density structure even when presented with small sample sizes at testing time. Similarly to the Neural Process [1], our method is able to construct a task embedding. In our case, however, embedding of each task takes the form of a conditional mean embedding operator, computed with feature maps learned using noise contrastive estimation. Further study could involve other choices of fake distribution $p_f(y)$, including those depending on the conditioning variable. An interesting avenue of applications would be in modelling conditional distributions in the reinforcement learning setting. In particular, [18] and [19] have shown the benefits of using distributional perspective on reinforcement learning as opposed to only modelling expectations of returns received by the agents.

References

[1] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

[2] Marta Garnelo, Dan Rosenbaum, Chris J Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J Rezende, and SM Eslami. Conditional neural processes. *arXiv preprint arXiv:1807.01613*, 2018.

[3] Le Song, Kenji Fukumizu, and Arthur Gretton. Kernel embeddings of conditional distributions: A unified kernel framework for nonparametric inference in graphical models. *IEEE Signal Processing Magazine*, 30(4):98–111, 2013.

[4] Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, Bernhard Schölkopf, et al. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1-2):1–141, 2017.

[5] Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(Feb):307–361, 2012.

[6] Saulius Gražulis, Adriana Daškevič, Andrius Merkys, Daniel Chateigner, Luca Lutterotti, Miguel Quiros, Nadezhda R Serebryanaya, Peter Moeck, Robert T Downs, and Armel Le Bail. Crystallography open database (cod): an open-access collection of crystal structures and platform for world-wide collaboration. *Nucleic acids research*, 40(D1):D420–D427, 2011.

[7] Brian L Trippe and Richard E Turner. Conditional density estimation with bayesian normalising flows. *arXiv preprint arXiv:1802.04908*, 2018.

[8] Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1751–1758, 2012.

[9] Zhuang Ma and Michael Collins. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. *arXiv preprint arXiv:1809.01812*, 2018.

[10] Aaron Van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[11] Bo Dai, Hanjun Dai, Arthur Gretton, Le Song, Dale Schuurmans, and Niao He. Kernel exponential family estimation via doubly dual embedding. *arXiv preprint arXiv:1811.02228*, 2018.

[12] Michael Arbel and Arthur Gretton. Kernel conditional exponential family. *arXiv preprint arXiv:1711.05363*, 2017.

[13] Yutian Chen, Max Welling, and Alex Smola. Super-samples from kernel herding. *arXiv preprint arXiv:1203.3472*, 2012.

[14] Motonobu Kanagawa. Empirical representations of probability distributions via kernel mean embeddings. 2016.

[15] Masashi Sugiyama, Ichiro Takeuchi, Taiji Suzuki, Takafumi Kanamori, Hirotaka Hachiya, and Daisuke Okanohara. Least-squares conditional density estimation. *IEICE Transactions on Information and Systems*, 93(3):583–594, 2010.

[16] Paul C. D. Hawkins. Conformation Generation: The State of the Art. *Journal of Chemical Information and Modeling*, 57(8):1747–1756, 2017. PMID: 28682617.

[17] Kanti V. Mardia. Statistical approaches to three key challenges in protein structural bioinformatics. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 62(3):487–514, 2013.

[18] Clare Lyle, Pablo Samuel Castro, and Marc G Bellemare. A comparative analysis of expected and distributional reinforcement learning. *arXiv preprint arXiv:1901.11084*, 2019.

[19] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 449–458. JMLR. org, 2017.

[20] Ruixiang Zhang, Tong Che, Zoubin Ghahramani, Yoshua Bengio, and Yangqiu Song. Metagan: An adversarial approach to few-shot learning. In *Advances in Neural Information Processing Systems*, pages 2365–2374, 2018.

Appendices

A On the choice of the fake distribution in NCE

The choice of the fake distribution plays a key role in the learning process here, especially due to the fact that we are interested in conditional densities. In particular, if the fake density is different from the marginal density $p(y)$, then our model could learn to distinguish between the fake and true samples of y simply by constructing a "good enough" model of the marginal density $p(y)$ on a given task while completely ignoring the dependence on x (this can be achieved by making the feature maps of x constant). This becomes obvious if, say, the supports of the fake and the true marginal distribution are disjoint, where clearly no information about x is needed to build a classifier – i.e. the classification problem is "too easy". Thus, ideally we wish to draw fake samples from the true marginal $p(y)$ in a given task. While we could achieve this by drawing a y paired to another x, i.e. from the empirical distribution of pooled ys in a given task, recall that we also require existence of a fake density which can be computed pointwise and inserted into (3). Hence, we propose to use a kernel density estimate (KDE) of ys as our fake density in any given task. In particular, kernel density estimator of $p(y)$ is computed on all responses y (context and target). In order to sample from the this fake distribution, we simply draw from the empirical distribution of pooled y's and add Gaussian noise with standard deviation being the bandwidth of the KDE (assuming we are using a Gaussian KDE for simplicity here; other choices of kernel are of course possible with appropriate modification of the type of noise). As our experiments demonstrate, this choice ensures that the fake samples are sufficiently hard to distinguish from the true ones, requiring the model to learn meaningful feature maps which capture the dependence between x and y and are informative for the CDE task.

Finally, we note that while in principle it is possible to consider families of fake distributions which also depend on the conditioning variable x, we do not explore this direction here. This is due to the fact that such approach would require a nontrivial construction of a model of fake conditional densities that is easy to sample from, can be computed pointwise, and according to the same rationale as above, shares the same marginal density with the true conditional model we are interested in.

B More details on related work

NCE for learning representations has been considered before and the closest work to our paper is [8][9], which focuses on learning discrete distributions in the context of Natural Language Processing (NLP). They achieve impressive speedups over other word embeddings as they avoid having to compute the normalizing constant thanks to the NCE setup of the optimization. More recently, [10] also introduce a NCE method for representation learning, however, they focus on learning an expressive representation in the unsupervised setting, thereby optimizing a mutual information objective instead.

Other methods that also use the idea of fake examples in order to learn an expressive feature map are [20], who train a GAN in order to use the resulting discriminator for few-shot classification.

In terms of using RKHSs in density models, several works, for example [11], [12] have considered training kernel exponential family models, where the main bottleneck is to compute the normalizing constant. [11] exploit the flexibility of kernel exponential families to learn conditional densities and avoid the problem of computing normalizing constants by solving so called nested Fenchel duals. [12] train kernel exponential family models using score matching criteria, which allows them to bypass normalizing constant computation. The method however requires computing and storing the first- and second order derivatives of the kernel function for each dimension and each sample and as such requires $\mathcal{O}(n^2d^2)$ memory and $\mathcal{O}(n^3d^3)$ time, where n is the number of data points and d the dimension of the problem. In addition, there has also been work done using CME for sampling methods, such as kernel herding, as presented in [13][14]. These methods are related in the sense that they also make use of the CME, however we use the CME as a feature to model the conditional density, whereas herding methods are interested in producing representative samples using the CME as a reference.

[15] propose a method of learning the conditional density by learning a ratio of the joint and the marginal. They model the conditional density as a linear combination of a set of basis functions. This

method works well on reasonably complicated tasks, although the optimal choice of basis functions is still unclear.

C Synthetic dataset setup and further experiments

In order to measure the ability of the method in learning the multimodality and heteroscedasticity in the response variable, we construct the datasets as follows: we sample $y_i \sim \text{Uniform}(0, 1)$ and set $x_i = \cos(ay_i + b) + \epsilon_i$, where a and b vary between tasks, with noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. Here we sample a from $U(8, 12)$ and phase vary from $U(0, \pi)$. In this experiment we are given a variable number of context points during testing time ranging from $15, 30$ and 50. Each of the non meta learning models DDE, KCEF, ϵ-KDE, LSCDE are trained on the new datasets. Our MetaCDE/Neural Process[1]/MetaNN is trained with $15, 30$ and 50 context points on the tasks respectively and with 80 target points. At testing time, we simply pass the data through our model without having to retrain on the new unseen dataset. Note that we report again the p-values of the Wilcoxon signed one-sided test and we can see that as we decrease the context points, our methods is significantly outperforming the other methods.

C.1 Model specifications

For our MetaCDE we used a 3-hidden layer Neural Network with $tanh$ activation functions and $Adam$ optimizer for all of our feature maps ϕ_x, ϕ_y, b_θ. We cross-validate on held out dataset, over 32 and 64 hidden nodes per layer and $\lambda = 1.0, 0.1, 0.01$ for the regularization parameter. We fixed the learning rate at 1e-3. We also set $\kappa = 10$.

- KCEF: we used the CV function that was in built in their Github repository
- LSCDE: We CV for σ in logspace$(-3, 5, 20)$ and λ in logspace$(-5, 5, 20)$
- ϵ-KDE: We CV over ϵ in linspace$(0.1, 1, 15)$ and bandwidth in linspace$(0.01, 1, 15)$
- DDE: We CV over the bandwidth of 0.5 and 1.0

D Neural Network version of our method (MetaNN)

In order to investigate the importance of the CMEO, in our setup we have tried to remove the operation completely and replaced it with a simple MLP. we do this by first of all concatenating the context pairs (x_i, y_i) into a vector, which we then pass through a MLP. At the end we take the mean of these vectors and concatenate them with the new target x_{target}. Hence we replaced the operation in Eq (5) with a NN. The rest stays the same. This method is hence also using meta-learning and therefore we can now compare against it in our experiments. In our experiments we simply used a 3 layer MLP and cross-validate over either 32, 64 hidden nodes and learning rates of 1e-3 or 1e-4.

D.1 Comparison to MetaCDE

Next we will compare the MetaNN and MetaCDE algorithms in order to investigate the importance of the conditional mean embedding operator. As mentioned above, we have now swapped out the computation of the CMEO for a NN. This representation is then concatenated with the new x_{target} to give us an element in $\mathcal{H}_Y$. We test out MetaNN on the synthetic dataset described in the experimental section. We start by highlighting that if use the experimental setup as described in Appendix C. MetaNN will perform worse on 50 context points as show in the main text but better on 30 and 15. MetaNN achieves a log-likelihood of 227.44 ± 41.92 and 114.43 ± 26.44 for 30 and 15 context points resepectively, where as MetaCDe achieved only 113.27 ± 17.27 and 51.73 ± 10.43. This can be explained by 2 factors. Firstly, a lower number of context points might give us a worse estimate of the conditional mean embedding operator. Secondly, we note that it takes significantly more task examples for the MetaNN to achieve the performance and hence this might have been due to the limited variety in the training task *i.e.* variation in the range of the period and phase parameter. Hence we conjectured that MetaNN might have just memorized the tasks well.

[1] We useed the following implementation `https://github.com/EmilienDupont/neural-processes`

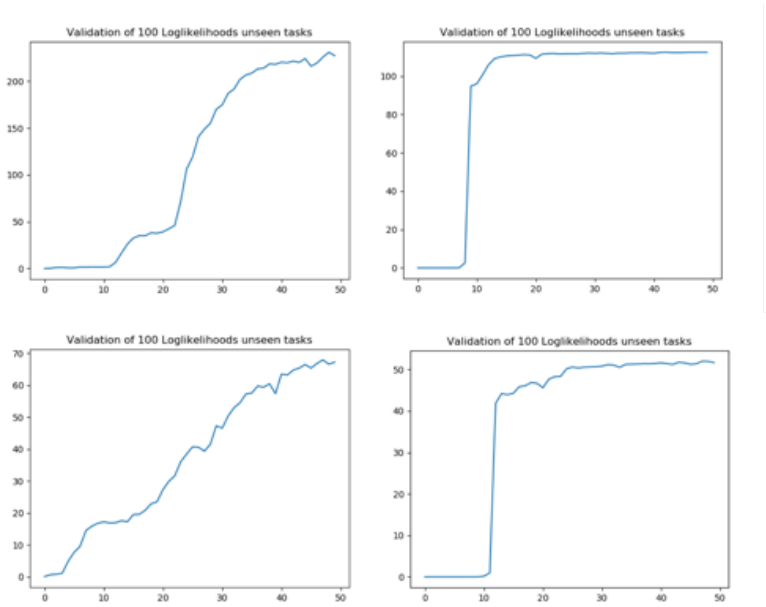

Figure 3: Figure illustrating how MetaNN performs better in low context points settings but seems to learn slower than MetaCDE.Top Row: 30 context points (left) MetaNN (right) MetaCDE; Bottom row: 15 context points (left) MetaNN (right) MetaCDE ((x-axis represents 1 unit=10k tasks))

Therefore we have ran additional experiments to on a harder synthetic dataset where we now sample a from $U(4, 14)$ and phase vary from $U(-\pi, \pi)$. In this case, MetaNN seems to completely fails and not able to learn anything useful at all. as the tasks in this case are more variable. Hence, we have not included MetaNN in the below figures when comparing with other conditional density methods.

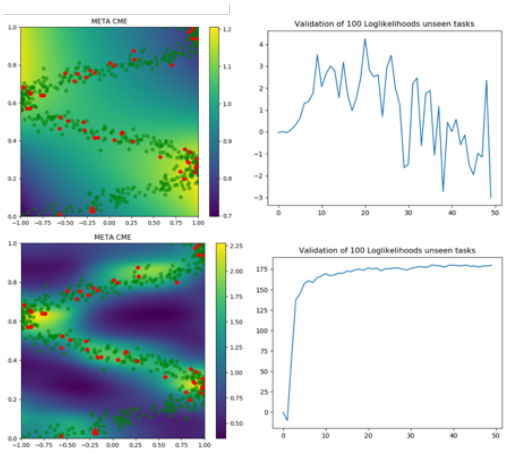

Figure 4: Figure illustrating how MetaNN fails when task become more variable. Top Row: MetaNN; Bottom row: MetaCDE ((x-axis represents 1 unit=10k tasks))

To further investigate this phenomenon, we have created a new task based on samples on Gaussian Processes (GPs). Here we sample 2 GPs with an Gaussian kernel with lenthscale 1 as well as a uniform random variable from $q \sim U(1, 3)$. We then added u to one of the sampled GPs and hence created a multimodal dataset in y(see figures below). This task has a lot more variability than the previous synthetic dataset task. Below, we illustrate how MetaCDE is still able tp perform well whereas MetaNN completely fails tp learn anything useful. This illustrates the useful and necessity of the CMEO which allows us to include additional inductive biases in our model. by using a CMEO,

we explicitly tell the model which entries are covariates and which are responses. Hence facilitating the learning process for the model.

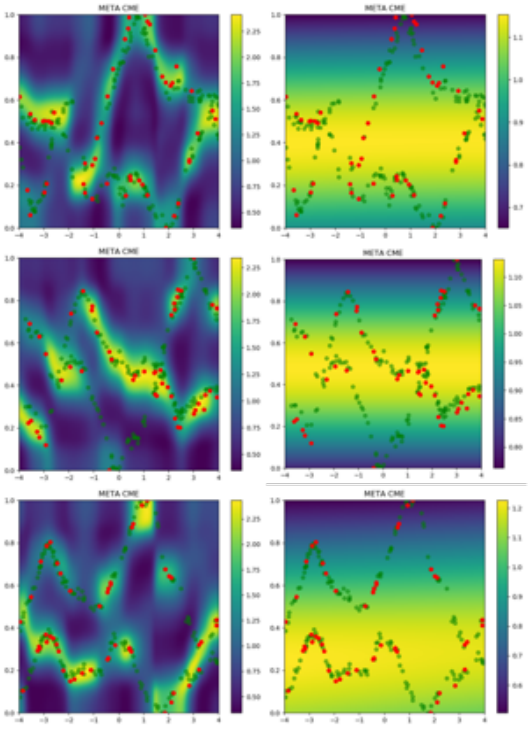

Figure 5: Density maps of the GP example (Left)MetaCDE (Right)(MetaNN)

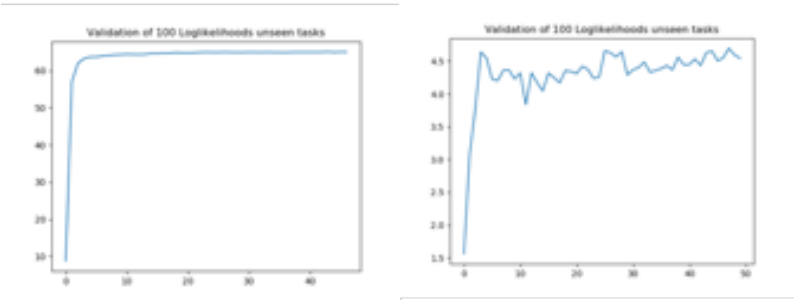

Figure 6: Evolution of the log-likelihood (x-axis represents 1 unit=10k tasks) (Left)MetaCDE (Right)(MetaNN)

E Illustration of Synthetic dataset

E.1 Using 50 context points

	MetaCDE	NP	DDE	LSCDE	KCEF	ϵ-KDE
Sythn. Data Mean over 100 log-likelihoods	**197.84** $\pm$ **22.45**	-81.11$\pm$18.53	162.98 $\pm$ 69.01	44.95 $\pm$ 74.36	-388.30 $\pm$ 703.17	116.31 $\pm$ 236.99
Sythn. Data P-value for Wilcoxon test	NA	< 2.2e-16	8.144e-07	<2.2e-16	< 2.2e-16	2.384e-07

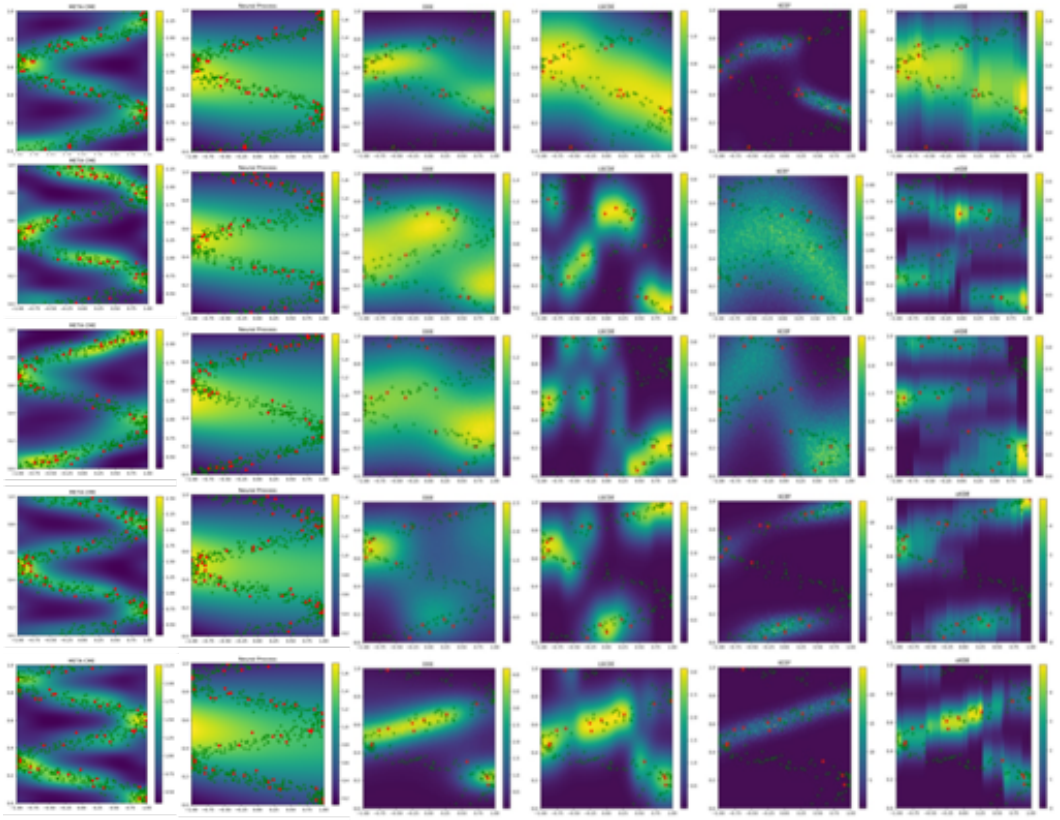

Figure 7: In Order (synthetic dataset): MetaCDE (ours), NP, DDE, LSCDE, KCEF, ϵ-KDE
The red dots are the context/training points and the green dots are points from the true density.

E.2 Using 30 context points

	MetaCDE	NP	DDE	LSCDE	KCEF	ϵ-KDE
Sythn. Data Mean over 100 log-likelihoods	**113.27 ± 17.36**	-48.98±12.26	64.61 ± 54.33	-23.02 ± 65.31	-233.38 ± 528.99	29.64 ± 195.49
Sythn. Data P-value for Wilcoxon test	NA	< 2.2e-16	4.577e-14	<2.2e-16	< 2.2e-16	4.917e-13

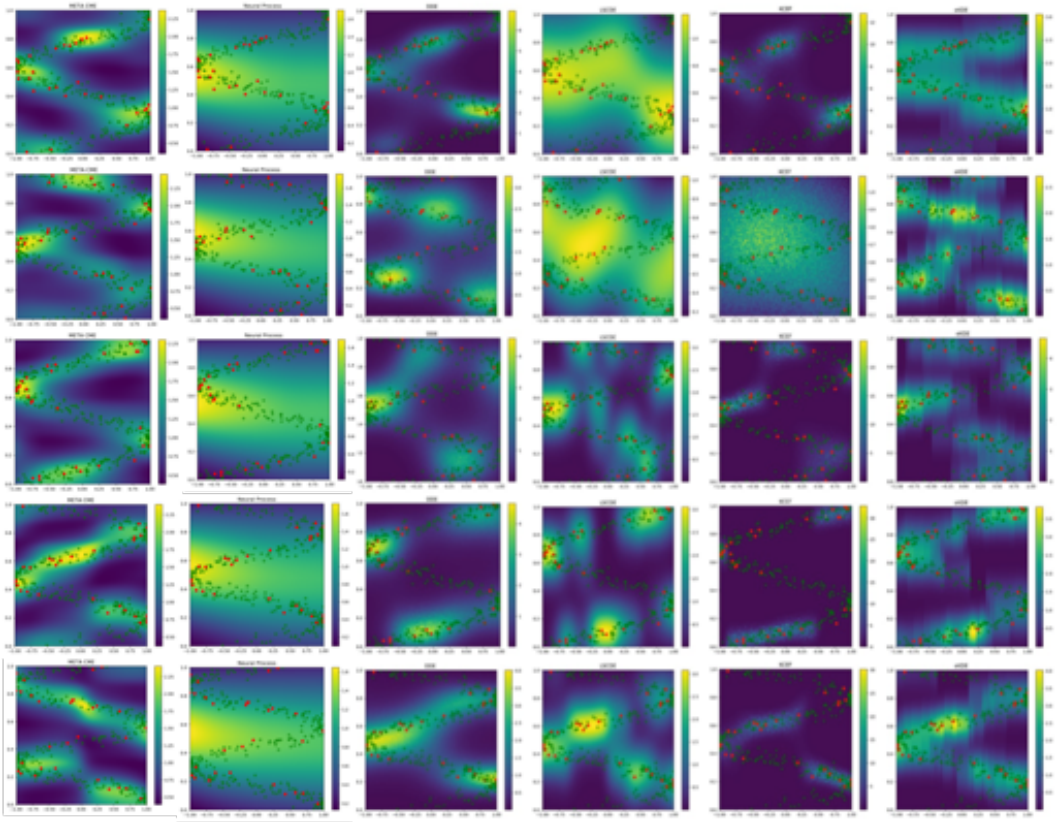

Figure 8: In Order (synthetic dataset): MetaCDE (ours), NP, DDE, LSCDE, KCEF, ϵ-KDE
The red dots are the context/training points and the green dots are points from the true density.

E.3 Using 15 context points

	MetaCDE	NP	DDE	LSCDE	KCEF	ϵ-KDE
Sythn. Data Mean over 100 log-likelihoods	**51.73 $\pm$ 10.48**	-24.39$\pm$8.20	0.58 $\pm$ 40.70	-57.99 $\pm$ 59.13	-142.19 $\pm$ 259.59	-87.50 $\pm$ 224.13
Sythn. Data P-value for Wilcoxon test	NA	< 2.2e-16	4.577e-14	<2.2e-16	< 2.2e-16	< 2.2e-16

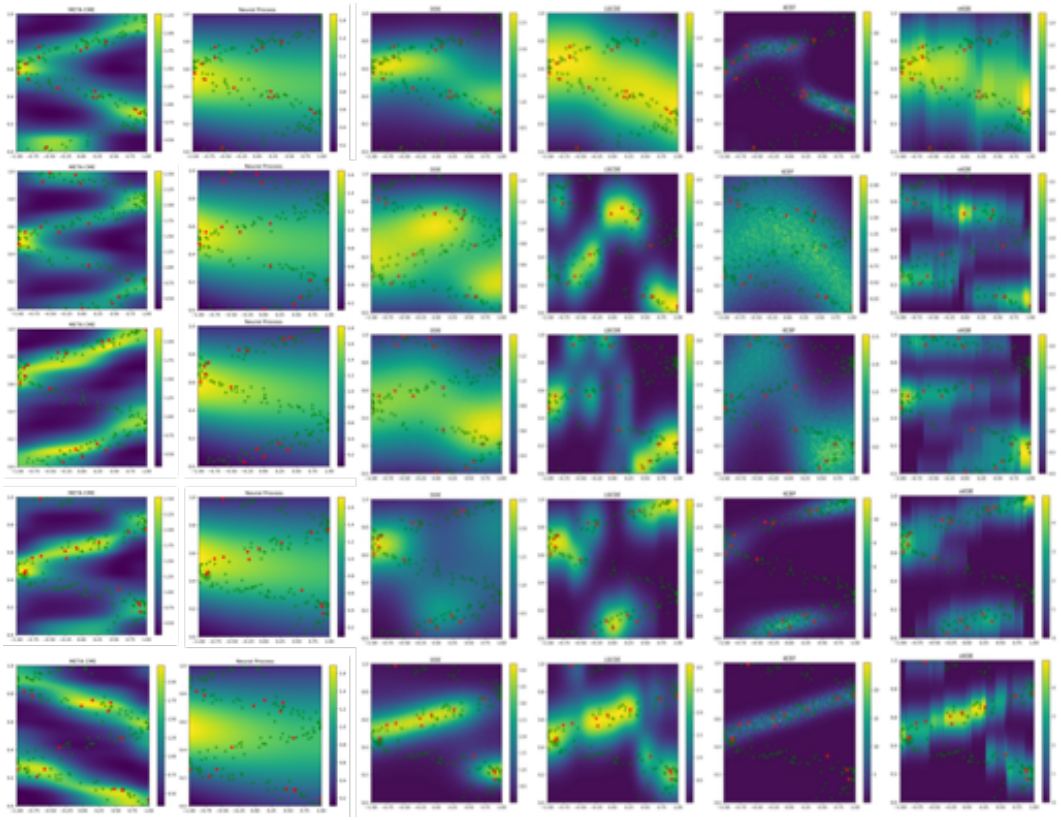

Figure 9: In Order (synthetic dataset): MetaCDE (ours), NP, DDE, LSCDE, KCEF, ϵ-KDE
The red dots are the context/training points and the green dots are points from the true density.

F Further insight to the Ramachandran plots

F.1 Additional information on the experimental setup

In this experiment, we look into the Ramachandran plots for molecules. Each plot indicates the energetically stable region of a pair of correlated dihedral angles in the molecule. Specifically, we are interested in estimating the distributions of these correlated dihedral angles. We compute the conditional density for each correlated dihedral angles, given 20 context points at testing time. For our meta-learning training we use 20 context points and 60 targets points.

Note that the data was extracted from crystallography database [6]. It is possible that some specific pairs of dihedral angles are rarely seen in the dataset, Hence, we may obtain a conditional density with high probability on the region without any observations in some cases. This is reasonable as the database covered only a small part of the chemical space and some potential area could be overlooked. Given that we assume that the support of our conditioning variable x ranges from $[-\pi, \pi)$, we will inevitable also compute conditional distribution on areas where the configurations are not defined and hence the densities in those areas can be safely ignored as a computational chemist would not have queried these configurations in the first place.

F.2 Model specifications

For our MetaCDE we used a 3 hidden layer NN with $tanh$ activation functions for all of our feature maps. We cross validate over 32 and 64 hidden nodes per layer and $\lambda = 1.0, 0.1$ for the regularization parameter. We fix the learning rate at $1e$-3 and set $\kappa = 10$.

- KCEF: we used the CV function that was in built in their Github repository
- LSCDE: We CV for σ in logspace$(-3, 5, 20)$ and λ in logspace$(-5, 5, 20)$
- ϵ-KDE: We CV over ϵ in linspace$(0.5, 3, 15)$ and bandwidth in linspace$(0.01, 3, 15)$
- DDE: We CV over bandwidth of 0.5 and 1.0

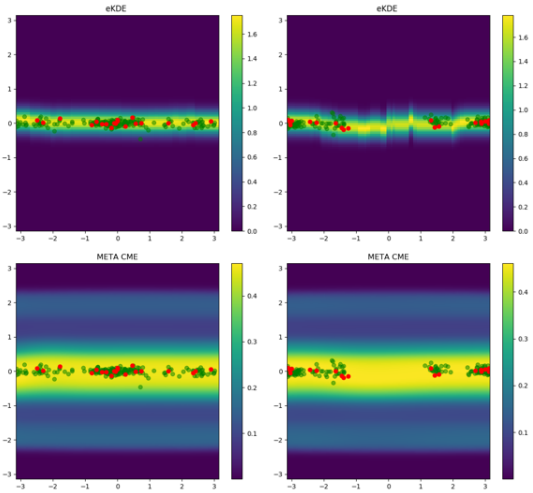

Figure 10: Cases where MetaCDE does not seem to perform better than conventional methods

Furthermore it looks like our method is not able to always capture the true trend given the limited amount of data. However, it seems to be able to capture some interesting patterns that would be useful to scientist to include in their models. Recently, there has been work done on these Ramachandran plots for Molecules but handcrafting the density maps. Our model would allow us to compute the density maps without prior knowledge.

G Illustration of the NYC taxi dataset

G.1 Experimental Setup

We have extracted the publicly available dataset from the website [2]. We have first of all restricted ourselves to drop-off locations in from -74.1 to -73.7 in longitude and 40.6 to 40.9 in latitude. Next we have given our meta learning model 200 datapoints for context during training and 300 for target. At testing time we are presented with 200 context points and are required to compute the conditional density given a tip. In this case each task is one specific pickup location. Again, we are using a 3-hidden layer NN with 128 nodes and CV over $\lambda = 0.1$ and 1.0. We use the $Adam$ optimizer and fixed the learning rate to $1\text{e} - 3$. We also set $\kappa = 10$.

G.2 Note on the dataset

In the main text we have seen how the drop-off density changes as we increase the amount of tips. This move of density illustrates well the data itself, as one is more likely to pay higher tips for longer journeys. Below we have plotted the drop-off locations of one specific pickup location colored with the respective tips paid.

[2] Data has been taken from: https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

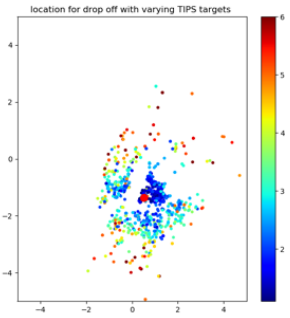

Figure 11: drop-off locations given a pickup location

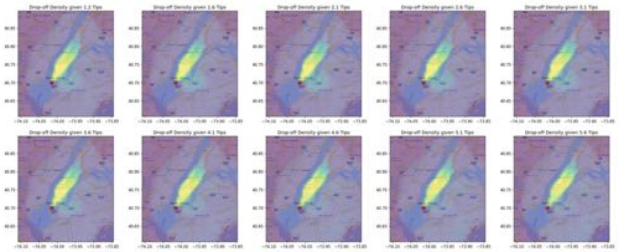

Figure 12: CDE of the drop-off locations as the tip amount increases. The trips starts at the red dot

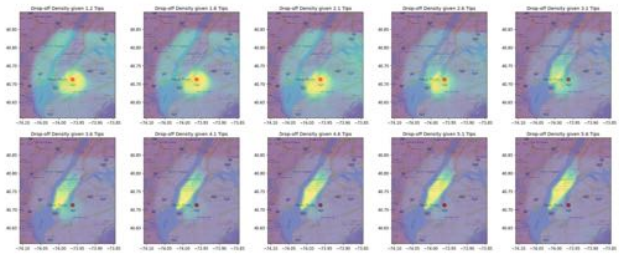

Figure 13: CDE of the drop-off locations as the tip amount increases. The trips starts at the red dot

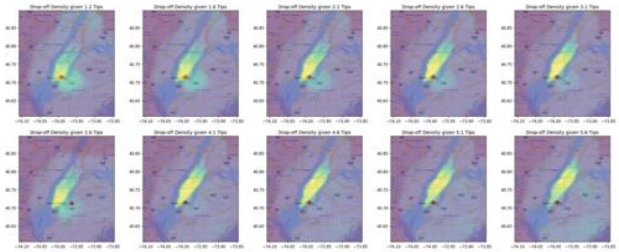

Figure 14: CDE of the drop-off locations as the tip amount increases. The trips starts at the red dot