Conditional density estimation using Fourier series and neural networks

M. H. de A. Inácio[1] and Rafael Izbicki[2]

[1] Universidade Federal de São Carlos e Universidade de São Paulo, Brazil
m@marcoinacio.com
[2] Universidade Federal de São Carlos, Brazil
rafaelizbicki@gmail.com

Abstract. Most machine learning tools aim at creating good predictions for new samples. However, obtaining 100% is not feasible in most problems, and therefore modeling the uncertainty over such predictions becomes necessary in several applications. This can be achieved by estimating conditional densities. In this work, we propose a novel method of conditional density estimation based on Fourier series and artificial neural networks, and compare it to other estimators on five distinct datasets. We conclude that our proposed method outperforms the other tested methods.

Categories and Subject Descriptors: G.3 [**Probability and Statistics**]: Nonparametric Statistics; I.2.6 [**Artificial Intelligence**]: Learning

Keywords: conditional density estimation, neural networks, Fourier series, pytorch

1. INTRODUCTION

Density estimation is among the most fundamental problems in Statistics. On the other hand, the field of Machine Learning and the so called algorithmic modeling has seen a recent surge in its popularity and applicability, due to, among other things, the vast amount of data available nowadays and a rapid increase of computational processing power.

However, this field has generally been more concerned with the problem of regression function (formally, the expected value $E[Y|x]$) than that of density estimation. In this work, we attempt to workout the problem of density estimation using a tool from such field that is also attracting great interest from researchers worldwide: artificial neural networks. More specifically, we show how artificial neural networks can be used to estimate a conditional density $f(y|x)$ in a fully nonparametric way. Conditional densities are more informative than regression functions: they model *all* uncertainty one has about Y given information x, and not only its expected value. Some well known methods of conditional density estimation are given by [Fan et al. 1996; Hall et al. 2004; Takeuchi et al. 2009; Efromovich 2010; Sugiyama et al. 2010]. Although such methods have good performance in several settings, they do not scale well to large datasets [Izbicki and Lee 2016; Izbicki and B. Lee 2017]; see [Bertin et al. 2016] and references therein for other methods.

The major contribution of this work is the proposal of a new method of conditional density estimation using neural networks which is highly scalable. In section 2, we introduce the problem of conditional density estimation. In section 3, we have a brief review of one of the tools that we use to work out such problem: feedforward artificial neural networks together with some recent advancements in the field which we take advantage to carry out this work. In section 4, we present a brief introduction to Fourier series, which is another tool that we use on our method of conditional density

estimation. We also briefly review some known methods of conditional density estimation that are useful to develop this work. In section 6, one can find empirical results of our proposed method using five datasets. Finally, section 7 concludes the article.

2. CONDITIONAL DENSITY ESTIMATION

Let $(X_1, Y_1), \ldots, (X_n, Y_n)$ be i.i.d. random vectors, where $Y_i \in \mathbb{R}$ is the response (label) and $X_i \in \mathbb{R}^d$ are covariates (features). Given that, problem of conditional density estimation can be stated simply as finding a good estimator $\hat{f}$ for the conditional density of $Y_k|X_k$, which we denote by $f(.|X_k) : [0,1] \to \mathbb{R} \in L_2[0,1]$, where $X_k = (X_{k1}, X_{k2}, ..., X_{kd})$. A simple solution for this problem is, for example, an ordinary least squares estimator:

$$\hat{f}(y|X_k) = \text{Gaussian}(X_k^T \hat{\beta}_{OLS}, \hat{\sigma}^2_{OLS})$$

Of course, such a simple estimator lacks flexibility for problems with complex structures both in terms of marginal density and in the structure of the covariates. Therefore, the goal of a good estimator is to be able to have considerable flexibility to model complex structure without incurring in excessive overfitting (bias-variance trade-off). In the next sections we review an already established method to deal with conditional density estimation using Fourier series, and introduce our proposed method which makes use of neural networks.

3. TOOLSET 1: NEURAL NETWORKS

Feedfoward artificial neural networks is one of the tools we make use of to carry out this work. In this section, we present its specification as follows:

Optimizer: we work with the Adamax optimizer ([Kingma and Ba 2014]) and decrease its learning rate if improvement is seen on the validation loss for a considerable number of epochs.

Initialization: we used the initialization method proposed by [Glorot and Bengio 2010].

Layer activation: we chose ELU ([Clevert et al. 2015]) as activation function.

Stop criteria: a 90%/10% split early stopping for small datasets and a higher split factor for larger datasets (increasing the proportion of training instances) and a patience of 50 epochs without improvement on the validation set.

Normalization and number of hidden layers: batch normalization, as proposed by [Ioffe and Szegedy 2015], is used in this work in order to speed-up the training process, specially since our networks have 10 hidden layers each.

Dropout: here we also make use of dropout which as proposed by [Hinton et al. 2012].

Software: we have PyTorch as framework of choice which works with automatic differentiation.

4. TOOLSET 2: FOURIER SERIES

4.1 Deterministic function approximation

Let $L^2([0,1])$ be the linear space of continuous functions $h_i : [0,1] \to \mathbb{R}$ such that $\int_0^1 h_i(x)\mathrm{d}x \leq \infty$ for $i \in \{1,2\}$. The usual inner product is defined by $\langle h_1, h_2 \rangle = \int_0^1 h_1(x)h_2(x)\mathrm{d}x$ and this inner product induces the following norm and distance in $L^2([0,1])$:

$$\|h_1\| = \left(\int_0^1 h_1^2(x)\mathrm{d}x \right)^{1/2} \quad \text{and} \quad \sqrt{\mathbb{M}(h_1, h_2)} = \left(\int_0^1 (h_1(x) - h_2(x))^2 \mathrm{d}y \right)^{1/2}$$

where $h_1, h_2 \in L^2([0,1])$. The sequence of functions $\{\phi_0, \phi_1, \phi_2, ...\}$ is called orthogonal system when $\langle \phi_i, \phi_j \rangle = 0$ for $i \neq j$ and $\|\phi_i\| \neq 0$ for all i. Furthermore, such system is called orthonormal basis if for any $h_1 \in L^2([0,1])$ there exists an unique sequence of scalars $\{\alpha_n\}_{n \in \mathbb{N}_+}$ such that $\left\| h_1 - \sum_{k=1}^{I} \alpha_k \phi_k \right\| \to 0$ as $I \to \infty$. Also, as of theorem 3.5.2 from [Kreyszig 1989], we have $\alpha_k = \langle h_1, \phi_i \rangle$.

Thus, h_1 has the following series representation given by $\sum_{i=0}^{\infty} \langle h_1, \phi_i \rangle \phi_i$.

Here, we shall consider the Fourier basis where $\phi_i : [0,1] \to [-\sqrt{2}, \sqrt{2}]$ and

$$\phi_i(x) = \begin{cases} 1 & \text{if } i = 0 \\ \sqrt{2}\sin(\pi(i+1)x) & \text{if } i \in \{1, 3, 5, ...\} \\ \sqrt{2}\cos(\pi i x) & \text{if } i \in \{2, 4, 6, ...\} \end{cases}$$

4.2 Non-conditional density estimation

Given i.i.d. random variables $Y_1, Y_2, ..., Y_n$ with density function $f : [0,1] \to \mathbb{R} \in L_2[0,1]$, then:

$$\hat{f}_I(y) = 1 + \sum_{i=1}^{I} \widehat{\alpha}_i \phi_i(y) \quad \text{with} \quad \widehat{\alpha}_i = \frac{1}{n} \sum_{j=1}^{n} \phi_i(Y_j) \approx \int \phi_i(y) f(y) dy = \langle \phi_i, f \rangle$$

is an approach to infer f from a frequentist perspective using Fourier series. Here, the choice of the estimator cutoff parameter I can be seen as bias-variance trade-off problem (in practice, a possible solution is to use cross-validation or data splitting to choose I).

However the estimate from $\hat{f}_I$ might not respect the constraint $\forall y \in [0,1], f(y) \geq 0$, in which case a "surgery" method is necessary (see [Wasserman 2006] and [Glad et al. 2003]). In case of Bayesian non-conditional density estimation, we have to define and work with priors in a constrained space where $f(y) \geq 0$ for all $y \in [0,1]$.

One way to overcome this issue is to use the approach of sieve priors suggested by [Scricciolo 2006] and applied in [Inacio et al. ress], which places a prior directly on the coefficient vector β of the Fourier series expansion of $\log(f)$ (instead of f) so that conditionally on the threshold parameter (cutoff parameter) I we have:

$$f(y|I, \beta) = \frac{1}{g(\beta, I)} \exp\left\{ \sum_{i=1}^{I} \beta_i \phi_i(y) \right\}$$

where g is a normalizing factor such that $g(\beta, I) = \int_0^1 \exp\left\{ \sum_{i=1}^{I} \beta_i \phi_i(y) \right\} \mathrm{d}y$. This is necessary in order to have $\int_0^1 f(y|I, \beta) \mathrm{d}y = 1$. Note that each β_i lives in $\mathbb{R}$, solving the constrained space problem.

As a drawback, we introduced the difficulty of calculating a normalizing factor (using numerical integration) when evaluating the likelihood function. As we shall see later, we will use a similar aproach to force $f(y) \geq 0$ for our proposed method.

4.3 Conditional density estimation: Flexcode

The Flexcode estimator ([Izbicki and B. Lee 2017]) is a natural extension of the frequentist density estimator method of section 4.2 to the conditional case and is used in this article as baseline to compare the results of our method. It consists in two steps:

(1) Estimate a regression function $r : \mathbb{R}^d \to [-\sqrt{2}, \sqrt{2}]^I$ where $r(.) = (r(.)_1, r(.)_2, ..., r(.)_I)$ and with $\phi_1(Y), \phi_2(Y), ..., \phi_I(Y)$ as targets and X as covariates. Such regression function can be obtained using a well known methods such as OLS, Lasso or KNN.

(2) Use the estimated regression to obtain following density estimate:

$$\widehat{f}(y_k|x_k) = 1 + \sum_{i=1}^{I} r(X)_i \phi(Y_k)$$

To understand why this procedure works, first notice that

$$f(y_k|x_k) = 1 + \sum_{i=1}^{\infty} \left(\int_0^1 \phi_i(y) f(y|x_k) \mathrm{d}y \right) \phi(y_k) = 1 + \sum_{i=1}^{\infty} E(\phi_i(Y)|x_k)\phi(y_k)$$

and that the fitted value $r(X_k)_i$ of a regression of $\phi_i(Y)$ against X is itself an estimate of $E(\phi_i(Y)|X_k)$. It follows that the choice of a cut-point I is a problem of bias-variance trade-off (in similar fashion to the non-conditional density estimator in 4.2) and, that in practice, this can be solved by cross-validation or data splitting.

5. OUR PROPOSED METHOD: CDFSNET

Our approach builds on Flexcode in order to achieve better performance and scalability. In our initial tests, we directly applied the Flexcode strategy to neural networks. That is, we trained a Neural network M with:

Input: a row vector input $x_k = (x_{k1}, x_{k2}, ..., x_{kd})$ of length d.
Output: a row vector $(M(x_k)_1, M(x_k)_2, ..., M(x_k)_I)$ of length I.

where the estimated density given by

$$\widehat{f}(y_k|x_k) = 1 + \phi_1(y_k)M(x_k)_1 + ...\phi_I(y_k)M(x_k)_I$$

and the loss on the training set is given by

$$\sum_{i=1}^{n}\sum_{j=1}^{I}(\phi_j(y_i) - M(x_i)_j)^2$$

However, this bare bones Flexcode procedure has shown to perform poorly on neural networks, even after applying the various neural networks techniques to avoid overfitting and local minimum convergence that we described in section 3 and even after attempting two different "surgery" methods (in order to force the estimated densities to be positive and integrate to 1).

On the order hand, instead of calculating a squared error on the regression (as in step 1 of Flexcode), we can work directly with the loss function and we can also apply a exponential transformation in similar fashion to what is proposed in 4.2 (i.e.: calculate the Fourier components of $\log(f)$ instead of f). In this case, the performance increases dramatically. Therefore we have a Neural network N with:

Input: a row vector input $x_k = (x_{k1}, x_{k2}, ..., x_{kd})$ of length d.
Output: a row vector $(N(x_k)_1, N(x_k)_2, ..., N(x_k)_I)$ of length I.

where the estimated density given by

$$\widehat{f}(y_k|x_k) = \frac{\exp\{\sum_{i=1}^{I} \phi_i(y_k)N(x_k)_i\}}{g(N(x_k), I)}$$

and where $g(N(x_k, I))$ is a normalizing factor. Here, we work with the integrated squared distance between the true and estimated density functions as loss function:

$$\int_\chi \int_0^1 (f(y|x) - \widehat{f}(y|x))^2 \mathrm{d}y \mathrm{d}P(x).$$

This loss can be estimated by

$$n^{-1}\sum_{i=1}^{n}\int_0^1 (f(y|x_i)-\widehat{f}(y|x_i))^2\mathrm{d}y = n^{-1}\sum_{i=1}^{n}\int_0^1 \left((\widehat{f}(y|x_i))^2 - 2f(y|x_i)\widehat{f}(y|x_i)\right)\mathrm{d}y + k$$

where k is a constant. It follows then that the loss function on the training set is given by a numerical approximation to

$$n^{-1}\sum_{i=1}^{n}\left(\int_0^1 (\widehat{f}(y|x_i))^2\mathrm{d}y - 2\widehat{f}(y_i|x_i)\right) \tag{1}$$

where the integration can be estimated numerically using, for example, the trapezoidal rule.

Moreover, on preliminary tests we also considered the softplus transformation[1] which is defined by $\frac{1}{b}\log(1+\exp(b*x))$ where $b=1$ (PyTorch default[2]). Since softplus transformation lead to better performance than the exponential transformation for a fixed amount of Fourier Series components, we decided to use it instead. This implies that we are in fact calculating the Fourier basis components of $\mathrm{softplus}^{-1}(f)$ and have the estimated density given by

$$\widehat{f}(y_k|x_k) = \frac{\mathrm{softplus}\{\sum_{i=1}^{I}\phi_i(y_k)N(x_k)_i\}}{g'(N(x_k),I)}$$

where $g'(N(x_k),I)$ is another normalizing factor.

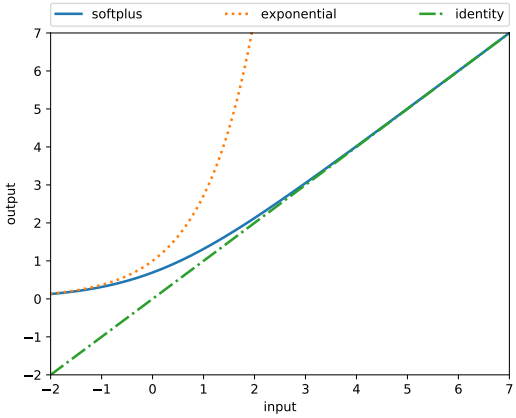

Fig. 1. Comparison of softplus, exponential and identity functions. Note that softplus function is always closer to the identity function that the exponential.

The intuition behind softplus giving some improvement over exponential is the fact that such transformation attempts to not significantly alter the value of its input (specially for large values), therefore potentially preserving the smoothness of the original (untransformed) density function. Figure 1 illustrates such property: softplus function is always to closer to the identity function that the exponential.

It is also worth noticing that this transformation and target loss function take advantage of the natural flexibility that neural networks have to minimize "arbitrary" loss functions and the speed GPUs can achieve when working with matrix multiplication which required for numerical integration inside the loss function. A Python package that implements the method that we therefore propose (and call CDFSNet) is available at `https://github.com/randommm/nncde`.

6. RESULTS AND ANALYSIS

We now present a comparison of CDFSNet and three implementations of Flexcode using five real world datasets that we describe in the subsections. The Python source code of these analysis is available at `https://github.com/randommm/nncde_implementation`.

6.1 Datasets and preprocessing

We compare FlexCode with CDFSNet on the following datasets:

[1]Softplus can also be used as an activation, but here we restrict its use as transformation to constrain the density function to positive values.
[2]We also tried other values for b, however the default value has shown to give better performance.

Spectroscopic dataset: We take spectroscopic data from [Izbicki and Lee 2016] and [Izbicki et al. 2017]. We take two subsets of the provided dataset. The first one with 10000 instances (in similar fashion to the one used by [Izbicki and Lee 2016]) and the second one with 100000. We work with the redshift income as the response variable.

Pnad dataset: We take a dataset from the Brazilian National Sample Survey of Households (PNAD), which is a research taken from Brazilian families and intends to extract information such as income, marriage, health, habitation and fecundity. For each attribute, we create an additional category to capture not available variables. We work with the family income as the response variable.

SGEMM dataset: We take a dataset from [Nugteren and Codreanu 2015] where the running time of a matrix-matrix product is measured, using a parameterizable SGEMM GPU kernel. For each combination of attributes, 4 runs were performed. For simplicity, we take the average of the 4 runs as the response variable.

Diamonds dataset: We take the classical diamonds dataset which is readily available from ggplot2 library and Kaggle. We work with carat as the response variable.

We use the following preprocessing in our experiments:

Response variable preprocessing: before training every model, we preproccess the response variable by taking its log and then transforming it to lie in the $(0, 1)$ interval. This is done for every dataset, with the exception of the spectroscopic dataset for which the response variable (the redshift of a galaxy) was already in the range of 0 to 1 in our received version.

Feature preprocessing: before training the neural networks, we preproccess all the features to have to mean 0 and variance 1 for all datasets, with the exception of the SGEMM dataset where we use PCA-Whitening transformation.

The score evaluation being carried out in a test dataset[3] (which was not used to train the models nor in the validation procedure for early stopping the neural networks).

6.2 Results

In Table I, we present the score (the opposite of the integrated squared distance loss as given by equation 1) of CDFSNet using the Flexcode implementation of nearest neighbors, XGBoosting and random forests[4] as the comparison baseline and using the aforementioned datasets.

Table I. Score (greater is better) of different methods for a given dataset. Here we compared Flexcode (using nearest neighbors, XGBoosting and random forests) with CDFSNet. NA represents a case where we were unable to train the model due to RAM limitations.

	Dataset				
	Spectroscopic		Pnad	SGEMM	Diamonds
Sample size	10000	100000	117939	241600	48940
Nº attributes	10	10	901	15	26
FC KNN	9.44	10.73	13.11	15.66	7.45
FC XGB	11.28	13.26	NA	18.86	15.56
FC RF	11.58	13.72	15.89	30.61	16.09
CDFSNet	13.57	16.63	15.95	54.74	19.79

In Figure 2, where we present the estimated conditional probability density function of the Fourier ANN and Flexcode Random Forest methods for the SGEMM dataset (conditional on a point chosen at random).

[3]The test dataset size was set to be the minimum between 5000 and 10% of the instances of the dataset.

[4]For such task, we used the Python Flexcode implementation available at `https://github.com/tpospisi/FlexCode` with the number of Fourier series components and some of the internal parameters of the estimators chosen by a data-splitting procedure.

6.3 Analysis

We notice from Table I that CDFSNet has outperformed all the other Flexcode based estimators. We note four possible reasons for such behaviour:

First, the neural network is trained to directly minimize the loss of interest, rather than several regression loss functions. Second, the Fourier series allows for a large number of Fourier series components to be used. Indeed, no hard cross-validation/data splitting procedure was necessary in order to choose the "cut point" of the Fourier series in our proposed method: a reasonably large neural network with 100 Fourier components works well "out-of-the-box" probably due to early stopping and dropout are already taking care of overfitting problems. On the other hand, for Flexcode estimators, a data splitting procedure generally dictates a much smaller number of Fourier series components due to the bias/variance tradeoff mentioned earlier.

Third, one of the limitations of the Flexcode method is that a Fourier series expansion might be negative in some regions, requiring some surgery procedures, and from Figure 2, we can see visually is that a large proportion of the density function is zeroed.

This also leads to a secondary effect of "stretching" the curve in points which already have positive density (in order force the density to integrate to 1). Intuitively, these effects may be causing an additional bias on the FlexCode density estimation for a given number of Fourier series components (a large number of Fourier series components might be able to overcome this issue, but at the price of larger variance). A theoretical study confirming this possibility is suggested as an extension of this work.

A fourth reason might be given by [Zhang et al. 2016] which discusses the capabilities that neural networks have in achieving generalization without falling into overfitting possibly due to properties of stochastic gradient descendant.

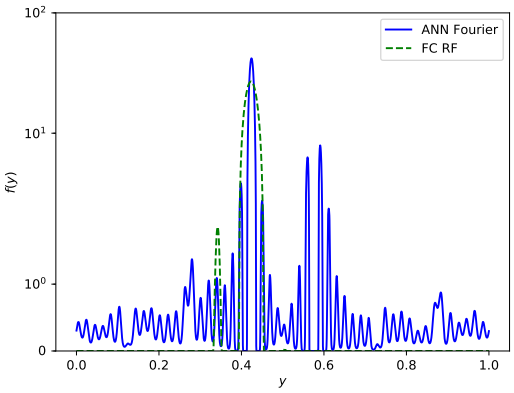

Fig. 2. Estimated probability conditional density function of the Fourier ANN and Flexcode Random Forest methods for the SGEMM dataset (conditional on a point chosen at random).

7. CONCLUSION

In this work, we have reviewed the concepts of Fourier series and conditional density estimation as well as an already established method of conditional density estimation using Fourier series. We have proposed a novel method of conditional density estimation that combines both Fourier series and artificial neural networks and compared it to the well established one using five datasets.

We have concluded that CDFSNet has outperforms the other tested methods while, in future works, we plan to explore how different architectures may lead to significantly better estimates of the conditional densities when dealing with non-standard data such as images and texts.

ACKNOWLEDGMENT

Marco Inacio is grateful for the financial support of CAPES; Rafael Izbicki is grateful for the financial support of FAPESP (grants 2014/25302-2 and 2017/03363-8) and CNPq (grant 306943/2017-4). We also thank Carlos Alberto Diniz, Roseli Romero and anonymous reviewers for their helpful comments on this work.

REFERENCES

Bertin, K., Lacour, C., and Rivoirard, V. Adaptive pointwise estimation of conditional density function. In *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*. Vol. 52. Institut Henri Poincaré, pp. 939–980, 2016.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus), 2015.

Efromovich, S. Dimension reduction and adaptation in conditional density estimation. *Journal of the American Statistical Association* 105 (490): 761–774, 2010.

Fan, J., Yao, Q., and Tong, H. Estimation of conditional densities and sensitivity measures in nonlinear dynamical systems. *Biometrika* 83 (1): 189–206, 1996.

Glad, I. K., Hjort, N. L., and Ushakov, N. G. Correction of density estimators that are not densities. *Scand J Stat* 30 (2): 415–427, jun, 2003.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. vol. 9, pp. 249–256, 01, 2010.

Hall, P., Racine, J., and Li, Q. Cross-validation and the estimation of conditional probability densities. *Journal of the American Statistical Association* 99 (468): 1015–1026, 2004.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, 2012.

Inacio, M., Izbicki, R., and Salasar, L. Comparing two populations using bayesian fourier series density estimation. *Communications in Statistics - Simulation and Computation*, in press.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei (Eds.). Proceedings of Machine Learning Research, vol. 37. PMLR, Lille, France, pp. 448–456, 2015.

Izbicki, R. and B. Lee, A. Converting high-dimensional regression to high-dimensional conditional density estimation. *Electron. J. Statist.* 11 (2): 2800–2831, 2017.

Izbicki, R. and Lee, A. B. Nonparametric conditional density estimation in a high-dimensional regression setting. *Journal of Computational and Graphical Statistics* 25 (4): 1297–1316, 2016.

Izbicki, R., Lee, A. B., and Freeman, P. E. Photo-z estimation: An example of nonparametric conditional density estimation under selection bias. *The Annals of Applied Statistics* 11 (2): 698–724, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR* vol. abs/1412.6980, 2014.

Kreyszig, E. *Introductory Functional Analysis with Applications*. Wiley, 1989.

Nugteren, C. and Codreanu, V. Cltune: A generic auto-tuner for opencl kernels. In *2015 IEEE 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip*. pp. 195–202, 2015.

Scricciolo, C. Convergence rates for Bayesian density estimation of infinite-dimensional exponential families. *The Annals of Statistics* 34 (6): 2897–2920, 2006.

Sugiyama, M., Takeuchi, I., Suzuki, T., Kanamori, T., Hachiya, H., and Okanohara, D. Conditional density estimation via least-squares density ratio estimation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. pp. 781–788, 2010.

Takeuchi, I., Nomura, K., and Kanamori, T. Nonparametric conditional density estimation using piecewise-linear solution path of kernel quantile regression. *Neural Computation* 21 (2): 533–559, 2009.

Wasserman, L. *All of nonparametric statistics*. Springer, New York London, 2006.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization, 2016.