Squared Neural Families of Tractable Densities

Dino Sejdinovic (University of Adelaide)

joint work with
Russell Tsuchida (Data 61 / CSIRO) and Cheng Soon Ong (Data 61 / CSIRO)

NeurIPS 2023, arXiv:2305.13552, github.com/RussellTsuchida/snefy

The Institute of Mathematical Statistics Asia-Pacific Rim Meeting
Melbourne, 5 January 2024

Deep Learning and Statistical Modelling

Deep Learning and Statistical Modelling

- Early approaches that combine deep learning and density modelling [LeCun et al., 2006] typically model energy functions as neural networks in Gibbs distributions – these distributions have intractable normalizing constants and require surrogate losses and various approximations.

Deep Learning and Statistical Modelling

- Early approaches that combine deep learning and density modelling [LeCun et al., 2006] typically model energy functions as neural networks in Gibbs distributions – these distributions have intractable normalizing constants and require surrogate losses and various approximations.

Deep Learning and Statistical Modelling

- Early approaches that combine deep learning and density modelling [LeCun et al., 2006] typically model energy functions as neural networks in Gibbs distributions – these distributions have intractable normalizing constants and require surrogate losses and various approximations.

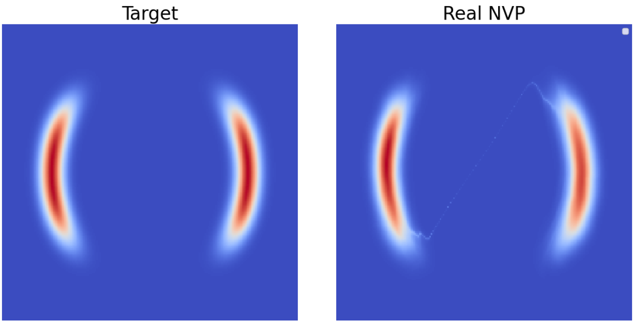
- An alternative is normalizing flows [Rezende and Mohamed, 2015] and its *many* variants, which use layers of bijective transformations based on neural networks with tractable Jacobians, starting with a simple distribution (i.e. Gaussian).

Deep Learning and Statistical Modelling

- Early approaches that combine deep learning and density modelling [LeCun et al., 2006] typically model energy functions as neural networks in Gibbs distributions – these distributions have intractable normalizing constants and require surrogate losses and various approximations.

- An alternative is normalizing flows [Rezende and Mohamed, 2015] and its *many* variants, which use layers of bijective transformations based on neural networks with tractable Jacobians, starting with a simple distribution (i.e. Gaussian).
- In this work, we propose a simple alternative to normalizing flows, which is fully tractable and has many advantageous properties.

The problem with normalizing flows



- For targets with supports / high density regions having complex topologies, flow must become arbitrarily close to noninvertible [Cornish et al., 2020, Caterini et al., 2021].
- The way around it is to consider an extended space: no longer a tractable density model and requires variational approximations for fitting!

Squared Neural Family (SNEFY)

- A measure space $(\Omega, \mathcal{F}, \mu)$, $\Omega \subseteq \mathbb{R}^d$,
- Domain $\mathbb{X} \in \mathcal{F}$,
- A mapping $\boldsymbol{t} : \mathbb{X} \to \mathbb{R}^D$ (sufficient statistic),
- One hidden layer neural network $\boldsymbol{f}(\boldsymbol{t}; \boldsymbol{V}, \Theta) = \boldsymbol{V}\sigma(\boldsymbol{W}\boldsymbol{t} + \boldsymbol{b})$
  - the activation function σ,
  - the hidden layer parameters $\Theta = (\boldsymbol{W}, \boldsymbol{b})$, with $\boldsymbol{W} \in \mathbb{R}^{n \times D}$ and $\boldsymbol{b} \in \mathbb{R}^n$,
  - the output layer parameters $\boldsymbol{V} \in \mathbb{R}^{m \times n}$.

Define a probability measure P such that its probability density p with respect to the base measure μ is proportional to the squared 2-norm of the neural network output

$$P(d\boldsymbol{x}; \boldsymbol{V}, \Theta) \triangleq \frac{\mu(d\boldsymbol{x})}{\mathrm{z}(\boldsymbol{V}, \Theta)} \left\| \boldsymbol{f}\left(\boldsymbol{t}(\boldsymbol{x}); \boldsymbol{V}, \Theta\right) \right\|_2^2.$$

Normalizing constant

$$
\begin{aligned}
z(\boldsymbol{V}, \Theta) &= \int_{\mathbb{X}} \left\| \boldsymbol{V} \sigma(\boldsymbol{W} \boldsymbol{t}(\boldsymbol{x}) + \boldsymbol{b}) \right\|_2^2 \mu(d\boldsymbol{x}) \\
&= \int_{\mathbb{X}} \sum_{i=1}^{n} \sum_{j=1}^{n} \boldsymbol{v}_{\cdot,i}^\top \boldsymbol{v}_{\cdot,j} \sigma\left(\boldsymbol{w}_i^\top \boldsymbol{t}(x) + b_i\right) \sigma\left(\boldsymbol{w}_j^\top \boldsymbol{t}(x) + b_j\right) \mu(d\boldsymbol{x}) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} \boldsymbol{v}_{\cdot,i}^\top \boldsymbol{v}_{\cdot,j} \left(\int_{\mathbb{X}} \sigma\left(\boldsymbol{w}_i^\top \boldsymbol{t}(x) + b_i\right) \sigma\left(\boldsymbol{w}_j^\top \boldsymbol{t}(x) + b_j\right) \mu(d\boldsymbol{x}) \right) \\
&= \operatorname{Tr}\left(\boldsymbol{V}^\top \boldsymbol{V} \boldsymbol{K}_\Theta \right),
\end{aligned}
$$

where $\boldsymbol{K}_\Theta$ is the PSD matrix whose ijth entry is

$$
k_{\sigma,t,\mu}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) := \int_{\mathbb{X}} \sigma\left(\boldsymbol{w}_i^\top \boldsymbol{t}(\boldsymbol{x}) + b_i\right) \sigma\left(\boldsymbol{w}_j^\top \boldsymbol{t}(\boldsymbol{x}) + b_j\right) \mu(d\boldsymbol{x}).
$$

Neural Network Kernels

- Classical work [Neal, 1995, Williams, 1997] connects infinite-width neural networks and Gaussian processes.
- Cho and Saul [2009] show that in some cases these kernels have a tractable form (e.g. ReLU activation gives rise to the arccosine kernel).
- Similar representations are used for large-scale approximations of kernel methods (e.g. random Fourier features of Rahimi and Recht [2007]).

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathbb{E}_{\mathrm{w}}\left[\sigma\left(\mathrm{w}^\top \boldsymbol{x}_i\right)\sigma\left(\mathrm{w}^\top \boldsymbol{x}_j\right)\right], \quad \mathrm{w} \sim \mathcal{N}\left(0, \boldsymbol{I}\right).$$

Neural Network Kernels

- Classical work [Neal, 1995, Williams, 1997] connects infinite-width neural networks and Gaussian processes.
- Cho and Saul [2009] show that in some cases these kernels have a tractable form (e.g. ReLU activation gives rise to the arccosine kernel).
- Similar representations are used for large-scale approximations of kernel methods (e.g. random Fourier features of Rahimi and Recht [2007]).

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathbb{E}_{\mathrm{w}}\left[\sigma\left(\mathrm{w}^\top \boldsymbol{x}_i\right)\sigma\left(\mathrm{w}^\top \boldsymbol{x}_j\right)\right], \quad \mathrm{w} \sim \mathcal{N}(0, \boldsymbol{I}).$$

- Different context: we wish to integrate with respect to inputs, not with respect to parameters!

$$k_{\sigma,t,\mu}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) := \int_{\mathbb{X}} \sigma\left(\boldsymbol{w}_i^\top \boldsymbol{t}(\boldsymbol{x}) + b_i\right)\sigma\left(\boldsymbol{w}_j^\top \boldsymbol{t}(\boldsymbol{x}) + b_j\right)\mu(d\boldsymbol{x}).$$

Neural Network Kernels

- Classical work [Neal, 1995, Williams, 1997] connects infinite-width neural networks and Gaussian processes.
- Cho and Saul [2009] show that in some cases these kernels have a tractable form (e.g. ReLU activation gives rise to the arccosine kernel).
- Similar representations are used for large-scale approximations of kernel methods (e.g. random Fourier features of Rahimi and Recht [2007]).

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathbb{E}_{\mathrm{w}}\left[\sigma\left(\mathrm{w}^\top \boldsymbol{x}_i\right)\sigma\left(\mathrm{w}^\top \boldsymbol{x}_j\right)\right], \quad \mathrm{w} \sim \mathcal{N}(0, \boldsymbol{I}).$$

- Different context: we wish to integrate with respect to inputs, not with respect to parameters!

$$k_{\sigma,t,\mu}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) := \int_{\mathbb{X}} \sigma\left(\boldsymbol{w}_i^\top \boldsymbol{t}(\boldsymbol{x}) + b_i\right)\sigma\left(\boldsymbol{w}_j^\top \boldsymbol{t}(\boldsymbol{x}) + b_j\right)\mu(d\boldsymbol{x}).$$

- In addition, bias terms will become important in this new context.

Examples of activation functions with tractable SNEFY

- Various examples that can be imported from the NNK literature:
  - erf
  - $(.)^p_+$, $p \in \mathbb{N}$, including ReLU
  - Leaky ReLU
  - GELU

  However, these settings are tractable only in the case $\boldsymbol{b} = 0$.

Examples of activation functions with tractable SNEFY

- Various examples that can be imported from the NNK literature:
  - erf
  - $(.)_+^p$, $p \in \mathbb{N}$, including ReLU
  - Leaky ReLU
  - GELU

  However, these settings are tractable only in the case $\boldsymbol{b} = 0$.
- New fully tractable settings:
  - $\sigma(u) = \exp(u/2)$: links to exponential families,
  - $\sigma(u) = \cos(u)$ and $\sigma(u) = \exp(iu)$: links to random Fourier features,
  - $\text{Snake}_a(u) = u + \frac{1}{a}\sin^2(au)$ which appears in the RL-based search for best activation functions [Ramachandran et al., 2018]

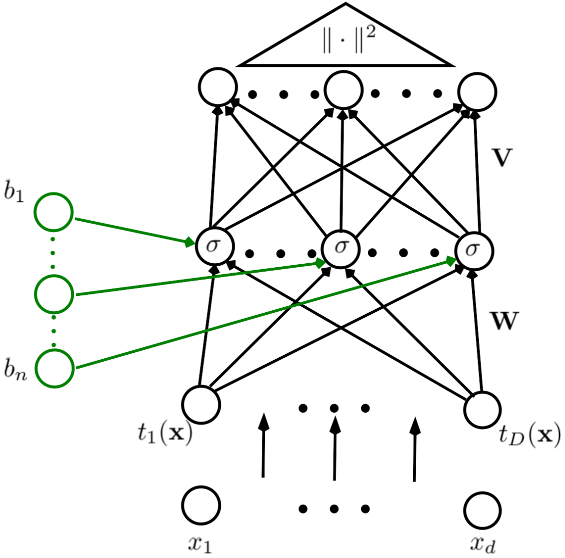
Examples of activation functions with tractable SNEFY

- Various examples that can be imported from the NNK literature:
  - erf
  - $(.)_+^p$, $p \in \mathbb{N}$, including ReLU
  - Leaky ReLU
  - GELU

  However, these settings are tractable only in the case $\boldsymbol{b} = 0$.
- New fully tractable settings:
  - $\sigma(u) = \exp(u/2)$: links to exponential families,
  - $\sigma(u) = \cos(u)$ and $\sigma(u) = \exp(iu)$: links to random Fourier features,
  - $\mathrm{Snake}_a(u) = u + \frac{1}{a}\sin^2(au)$ which appears in the RL-based search for best activation functions [Ramachandran et al., 2018]

Recall that what makes a particular SNEFY tractable is the triplet $(\sigma, \mathrm{t}, \mu)$ – refer to paper for details.

SNEFY Illustration



A deep SNEFY version?

- Unclear if there is a tractable case with multiple hidden layers (we haven't found any)

A deep SNEFY version?

- Unclear if there is a tractable case with multiple hidden layers (we haven't found any)

- There is however a simple way to make more expressive density models: use SNEFY as a base measure in a normalizing flow – the model remains fully tractable and can be trained end-to-end, avoiding the pathologies of complex topology.

Experiments: Unconditional case

	SNEFY 0	Gauss 0	GMM 0	SNEFY 16	Gauss 16	GMM 16
Moons	-1.59 ± 0.02	-3.29 ± 0.02	-1.59 ± 0.04	-1.57 ± 0.03	-1.68 ± 0.24	-1.57 ± 0.03
	241	4	50	19281	19044	19090
	1200.84 ± 109.77	606.35 ± 20.29	629.11 ± 14.57	2844.09 ± 254.02	2226.37 ± 174.80	2276.78 ± 211.15
Circles	-1.92 ± 0.03	-3.55 ± 0.01	-2.07 ± 0.07	-1.92 ± 0.04	-2.22 ± 0.30	-1.93 ± 0.03
	241	4	50	19281	19044	19090
	643.80 ± 150.84	52.63 ± 3.96	73.52 ± 5.47	2272.15 ± 281.76	1660.77 ± 169.02	1673.54 ± 166.84
Rings	-2.41 ± 0.05	-3.26 ± 0.01	-2.67 ± 0.03	-2.34 ± 0.10	-2.51 ± 0.22	-2.31 ± 0.04
	241	4	50	19281	19044	19090
	997.73 ± 107.80	416.23 ± 18.11	433.82 ± 16.06	2654.22 ± 269.14	2045.65 ± 170.80	2070.60 ± 177.06

Table:
The first quantity is the average $\pm$ sample standard deviation over 20 runs of *test loglikelihood*.
The second quantity is the *parameter count*.
The third quantity is the average $\pm$ sample standard deviation over 20 runs of *the computation time (seconds)*.
The number in each column header is the number of non-volume preserving flow layers appended after the base distribution. Here there are 0 or 16 NVP layers.

SNEFY generalizes the exponential family mixture models

Consider the activation

$$\sigma(u) = \exp(u/2).$$

In this case, we can absorb the bias terms $\boldsymbol{b}$ into the output layer parameters $\boldsymbol{V}$. We obtain the following family of distributions

$$P(d\boldsymbol{x}; \boldsymbol{V}, \boldsymbol{W}) = \frac{1}{\mathrm{Tr}(\boldsymbol{V}^\top \boldsymbol{V} \boldsymbol{K}_\Theta)} \sum_{i=1}^{n} \sum_{j=1}^{n} \boldsymbol{v}_{\cdot,i}^\top \boldsymbol{v}_{\cdot,j} \exp\left(\frac{1}{2}(\boldsymbol{w}_i + \boldsymbol{w}_j)^\top \boldsymbol{t}(\boldsymbol{x})\right) \mu(d\boldsymbol{x}),$$

which is a mixture of distributions $P_e\left(\cdot; \frac{1}{2}(\boldsymbol{w}_i + \boldsymbol{w}_j)\right)$ belonging to a classical exponential family P_e given in the canonical form by

$$P_e(d\boldsymbol{x}; \boldsymbol{w}) = \frac{1}{z_e(\boldsymbol{w})} \exp\left(\boldsymbol{w}^\top \boldsymbol{t}(\boldsymbol{x})\right) \mu(d\boldsymbol{x}), \quad z_e(\boldsymbol{w}) = \int_{\mathbb{X}} \exp\left(\boldsymbol{w}^\top \boldsymbol{t}(\boldsymbol{x})\right) \mu(d\boldsymbol{x}),$$

with potentially negative mixture weights $\boldsymbol{v}_{\cdot,i}^\top \boldsymbol{v}_{\cdot,j}$ defined by the output layer parameters $\boldsymbol{V}$.

SNEFY generalizes the exponential family mixture models

The kernel matrix $\boldsymbol{K}_\Theta$ in the normalizing constant of SNEFY is tractable whenever the normalizing constant of the corresponding exponential family is itself tractable since

$$k_{\exp(\cdot/2),t,\mu}(\boldsymbol{w}_i, \boldsymbol{w}_j) = z_e\left(\frac{1}{2}(\boldsymbol{w}_i + \boldsymbol{w}_j)\right),$$

where z_e is the normalising constant of the exponential family P_e.
Examples include

- SNEFY Von Mises-Fisher mixtures on the sphere,
- SNEFY Dirichlet mixtures on the simplex,
- SNEFY Gaussian mixtures which are equivalent to modelling a density using the square of a Radial Basis Function (RBF) network.

SNEFY generalizes the exponential family mixture models

The kernel matrix $\boldsymbol{K}_\Theta$ in the normalizing constant of SNEFY is tractable whenever the normalizing constant of the corresponding exponential family is itself tractable since

$$k_{\exp(\cdot/2),t,\mu}(\boldsymbol{w}_i, \boldsymbol{w}_j) = z_e \left(\frac{1}{2}(\boldsymbol{w}_i + \boldsymbol{w}_j) \right),$$

where z_e is the normalising constant of the exponential family P_e.
Examples include

- `SNEFY` Von Mises-Fisher mixtures on the sphere,
- `SNEFY` Dirichlet mixtures on the simplex,
- `SNEFY` Gaussian mixtures which are equivalent to modelling a density using the square of a Radial Basis Function (RBF) network.

Some properties of exponential families carry over: for $\Psi = \log z(\boldsymbol{V}, \Theta)$,

$$\sum_{i=1}^{n} \frac{\partial \Psi}{\partial \boldsymbol{w}_i} = \mathbb{E}\left[\boldsymbol{t}(\mathrm{x})\right], \quad \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 \Psi}{\partial \boldsymbol{w}_i \boldsymbol{w}_j^\top} = \mathbb{E}\left[\boldsymbol{t}(\mathrm{x})\,\boldsymbol{t}(\mathrm{x})^\top\right] - \mathbb{E}\left[\boldsymbol{t}(\mathrm{x})\right]\mathbb{E}\left[\boldsymbol{t}(\mathrm{x})\right]^\top.$$

Has nobody really noticed this before?

Has nobody really noticed this before?

- Surprisingly, no (to the best of our knowledge).

Has nobody really noticed this before?

- Surprisingly, no (to the best of our knowledge).

- The closest prior work is the non-parametric kernel models for non-negative functions [Marteau-Ferey et al., 2020],

$$h_{\mathsf{A}}(\boldsymbol{x}) = \langle \psi(\boldsymbol{x}), \boldsymbol{A}\psi(\boldsymbol{x}) \rangle_{\mathbb{H}},$$

  for a Hilbert space $\mathbb{H}$, feature map $\psi(\boldsymbol{x})$, and a PSD operator $\boldsymbol{A}$, but which can be normalized only in a very limited setting which turns out to be equivalent to our `SNEFY` Gaussian mixtures.

Properties

Let $\mathrm{x} = (\mathrm{x}_1, \mathrm{x}_2)$ be jointly $\texttt{SNEFY}_{\mathbb{X}_1 \times \mathbb{X}_2, t, \sigma, \mu}$ with parameters $\boldsymbol{V}$ and $\Theta = ([\boldsymbol{W}_1, \boldsymbol{W}_2], \boldsymbol{b})$. Assume $\mu(d\boldsymbol{x}) = \mu_1(d\boldsymbol{x}_1)\mu_2(d\boldsymbol{x}_2)$, $\boldsymbol{t}(\boldsymbol{x}) = \big(\boldsymbol{t}_1(\boldsymbol{x}_1), \boldsymbol{t}_2(\boldsymbol{x}_2)\big)$.

SNEFY **is closed under conditioning**

Theorem (Conditioning)

The conditional distribution of x_1 *given* $\mathrm{x}_2 = \boldsymbol{x}_2$ *is* $\texttt{SNEFY}_{\mathbb{X}_1, t_1, \sigma, \mu_1}$ *with parameters* $\boldsymbol{V}$ *and* $\Theta_{1|2} \triangleq (\boldsymbol{W}_1, \boldsymbol{W}_2 \boldsymbol{t}_2(\boldsymbol{x}_2) + \boldsymbol{b})$.

SNEFY **has tractable marginals**

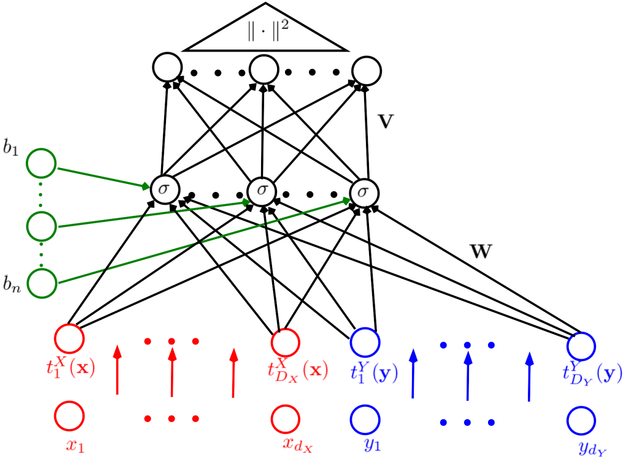
Theorem (Marginalization)

The marginal distribution of x_1 *is*

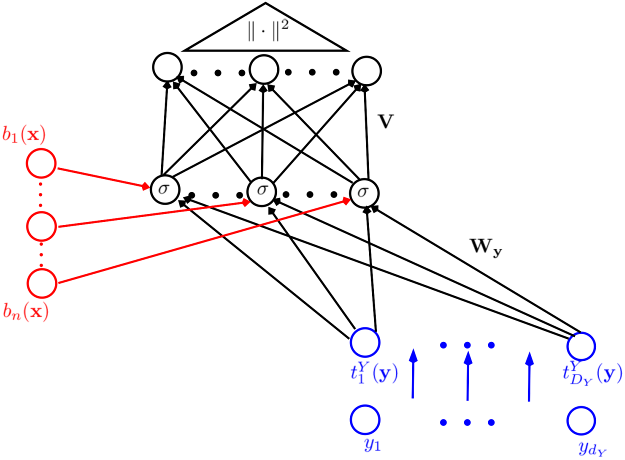
$$P_1(d\boldsymbol{x}_1) = \frac{\mathrm{Tr}\left(\boldsymbol{V}^\top \boldsymbol{V} \widetilde{\boldsymbol{C}}_\Theta(\boldsymbol{x}_1)\right)}{\mathrm{z}(\boldsymbol{V}, \Theta)} \mu_1(d\boldsymbol{x}_1),$$

where $\widetilde{C}(\boldsymbol{x}_1)_{ij} = k_{\sigma, t_2, \mu_2}\Big(\big(\boldsymbol{w}_{2i}, \boldsymbol{w}_{1i}^\top \boldsymbol{t}_1(\boldsymbol{x}_1) + b_i\big), \big(\boldsymbol{w}_{2j}, \boldsymbol{w}_{1j}^\top \boldsymbol{t}_1(\boldsymbol{x}_1) + b_j\big)\Big)$.

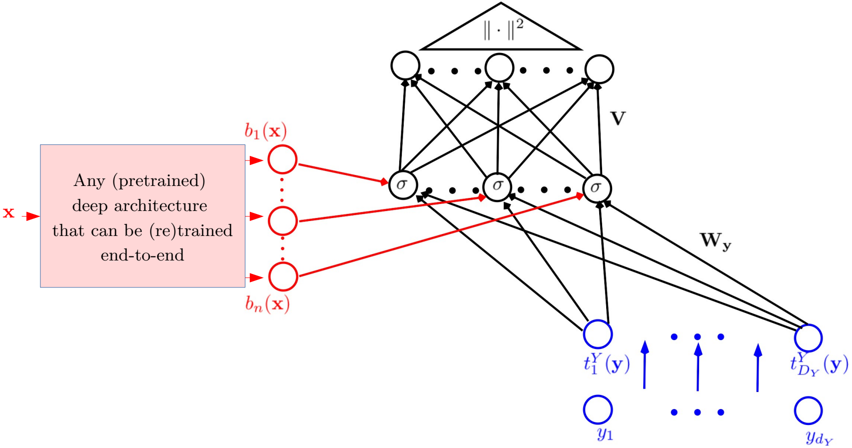
Conditional Density Estimation



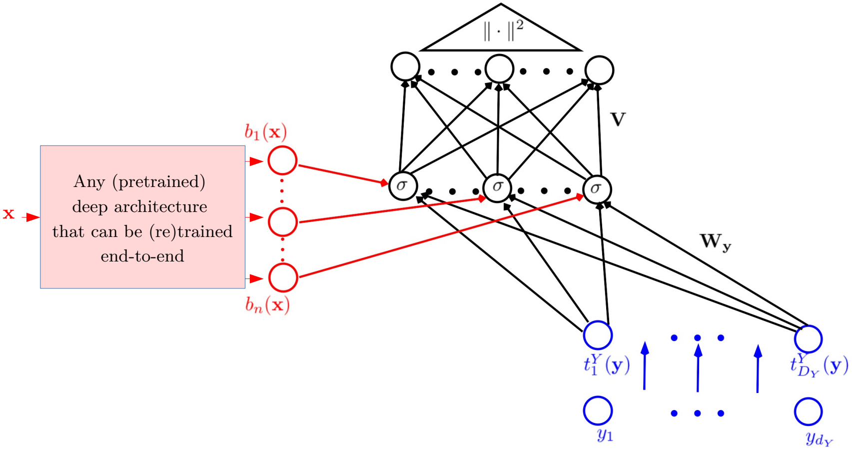
Conditional Density Estimation



Conditional Density Estimation



Conditional Density Estimation



Can turn any regression method (estimation of $\mathbb{E}[y|x]$) into tractable conditional density estimation (of $p(y|x)$).

Experiments: Conditional case

Method	Average Test log-likelihood $\uparrow$	Compute time (s)	Parameter count
SNEFY $n = 32$	2.195 ± 0.024	495.720 ± 22.561	179748
SNEFY $n = 16$	2.172 ± 0.034	404.291 ± 57.605	46356
SNEFY $n = 8$	2.108 ± 0.089	390.870 ± 16.028	12300
CNF $L = 4$	2.156 ± 0.018	202.1290 ± 10.380	1413
CNF $L = 2$	2.163 ± 0.024	155.090 ± 15.809	1155
CNF $L = 1$	2.171 ± 0.012	122.304 ± 1.194	1026
CKDE	2.148	391.867 (Not GPU-accelerated)	Train set size $= 74309 \times 6$

Table: Performance comparison of methods on astronomy dataset (photometric redshift). Excluding CKDE which is deterministic, an average is taken over 50 random initialisations. SNEFY shows a statistically significant average increase in performance over CNF, and over CKDE.

Experiments: Missing data

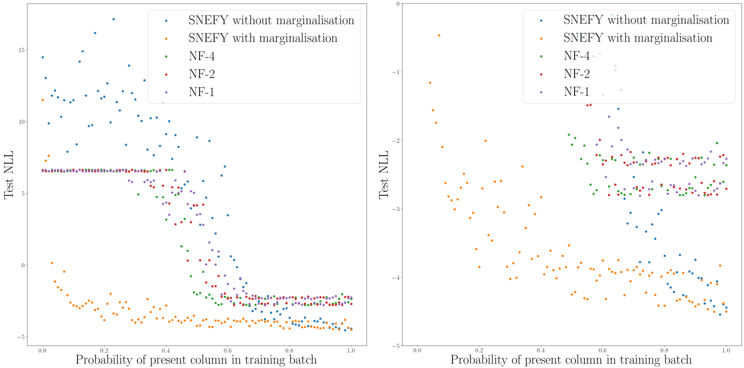


Figure: Density estimation under partial observations. The right plot is a zoomed in version of the left plot. NF-1, NF-2 and NF-4 are respectively normalising flows of depth 1, 2 and 4. Normalising flow models and `SNEFY` without marginalisation discard incomplete observations, whereas `SNEFY` use tractability of the marginal distribution to include partial observations via standard maximum marginal likelihood. Large improvements in held-out NLL for high missingness.

Discussion

- SNEFY is well suited for density estimation and conditional density estimation – fully tractable models which can be fitted using maximum likelihood end-to-end.
- Tractability holds even in case of missing observations.
- Flexibility to reuse existing deep architectures: a direct recipe to turn any regression method into conditional density estimation.
- Sampling from SNEFY: not obvious that there are shortcuts, and one needs to resort to rejection sampling or MCMC. Alternative approaches such as normalizing flows (with easy to sample base measures) may be better suited if sampling from fitted densities is required.
- Are there any fully tractable deep SNEFY cases?

References I

Russell Tsuchida, Cheng Soon Ong, and Dino Sejdinovic. Squared neural families: A new class of tractable density models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang. A tutorial on energy-based learning. *Predicting structured data*, 2006.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning (ICML)*, 2015.

Rob Cornish, Anthony Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. In *International Conference on Machine Learning (ICML)*, 2020.

Anthony Caterini, Rob Cornish, Dino Sejdinovic, and Arnaud Doucet. Variational inference with continuously-indexed normalizing flows. In *Uncertainty in Artificial Intelligence (UAI)*, 2021.

Radford M Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995.

References II

Christopher KI Williams. Computing with infinite networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1997.

Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2009.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2007.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2018. URL `https://openreview.net/forum?id=SkBYYyZRZ`.

Ulysse Marteau-Ferey, Francis Bach, and Alessandro Rudi. Non-parametric models for non-negative functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.