Surjectors: surjection layers for density estimation with normalizing flows

Simon Dirmeier[1,2]

1 Swiss Data Science Center, Zurich, Switzerland **2** ETH Zurich, Zurich, Switzerland

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Arfon Smith

Reviewers:

- @sandeshkatakam
- @animikhaich

Submitted: 14 October 2023
Published: unpublished

Summary

Normalizing flows (NFs, Papamakarios et al., 2021) are tractable neural density estimators which have in the recent past been applied successfully for, e.g., generative modelling Ping et al. (2020), Bayesian inference Hoffman et al. (2019) or simulation-based inference Dirmeier et al. (2023). `Surjectors` is a Python library in particular for *surjective*, i.e., dimensionality-reducing normalizing flows (SNFs, Klein et al. (2021)). `Surjectors` is based on the libraries JAX, Haiku and Distrax Babuschkin et al. (2020) and is fully compatible with them. By virtue of being entirely written in JAX (Bradbury et al., 2018), `Surjectors` naturally supports usage on either CPU, GPU or TPU.

Statement of Need

Real-world data are often lying in a high-dimensional ambient space embedded in a lower-dimensional manifold (Fefferman et al., 2016) which can complicate estimation of probability densities (Dai & Seljak (2021), Klein et al. (2021),Nalisnick et al. (2019)). As a remedy, recently neural density estimators using surjective normalizing flows (SNFs) have been proposed which reduce the dimensionality of the data while still allowing for exact computation of data likelihoods (Klein et al., 2021). While several computational libraries exist that implement *bijective* normalizing flows, i.e., flows that are dimensionality-preserving, currently none exist that efficiently implement dimensionality-reducing flows.

`Surjectors` is a normalizing flow library that implements both bijective and surjective normalizing flows. `Surjectors` is light-weight, conceptually simple to understand if familiar with the JAX ecosystem, and computationally efficient due to leveraging the XLA compilation and vectorization from JAX. We additionally make use of several well-established packages within the JAX ecosystem (Bradbury et al., 2018) and probabilistic deep learning community. For composing the conditioning networks that NFs facilitate, `Surjectors` uses the deep learning library Haiku (Hennigan et al., 2020). For training and optimisation, we utilize the gradient transformation library Optax (Babuschkin et al., 2020). `Surjectors` leverages Distrax (Babuschkin et al., 2020) and TensorFlow probability (Dillon et al., 2017) for probability distributions and several base bijector implementations.

Adoption

Dirmeier et al. (2023) have proposed a novel method for simulation-based inference where they make use autoregressive inference surjections for density estimation and where they are using `Surjectors` for their implementations.

References

Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., Buchlovsky, P., Budden, D., Cai, T., Clark, A., Danihelka, I., Dedieu, A., Fantacci, C., & others. (2020). *The DeepMind JAX Ecosystem*. http://github.com/deepmind

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs*. http://github.com/google/jax

Dai, B., & Seljak, U. (2021). Sliced iterative normalizing flows. *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*.

Dillon, J. V., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., Patton, B., Alemi, A., Hoffman, M., & Saurous, R. A. (2017). Tensorflow distributions. *arXiv Preprint arXiv:1711.10604*.

Dirmeier, S., Albert, C., & Perez-Cruz, F. (2023). Simulation-based inference using surjective sequential neural likelihood estimation. *arXiv Preprint arXiv:2308.01054*.

Fefferman, C., Mitter, S., & Narayanan, H. (2016). Testing the manifold hypothesis. *Journal of the American Mathematical Society*, *29*(4), 983–1049.

Hennigan, T., Cai, T., Norman, T., Martens, L., & Babuschkin, I. (2020). *Haiku: Sonnet for JAX*. http://github.com/deepmind/dm-haiku

Hoffman, M., Sountsov, P., Dillon, J. V., Langmore, I., Tran, D., & Vasudevan, S. (2019). Neutra-lizing bad geometry in hamiltonian monte carlo using neural transport. *arXiv Preprint arXiv:1903.03704*.

Kingma, D. P., & Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. *Advances in Neural Information Processing Systems*.

Klein, S., Raine, J. A., Pina-Otey, S., Voloshynovskiy, S., & Golling, T. (2021). Funnels: Exact maximum likelihood with dimensionality reduction. *Workshop on Bayesian Deep Learning, Advances in Neural Information Processing Systems*.

Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., & Lakshminarayanan, B. (2019). Do deep generative models know what they don't know? *International Conference on Learning Representations*.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., & Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, *22*(1), 2617–2680.

Papamakarios, G., Sterratt, D., & Murray, I. (2019). Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*.

Ping, W., Peng, K., Zhao, K., & Song, Z. (2020). WaveFlow: A compact flow-based model for raw audio. *Proceedings of the 37th International Conference on Machine Learning*.

Rezende, D., & Mohamed, S. (2015). Variational inference with normalizing flows. *Proceedings of the 32nd International Conference on Machine Learning*.