Generative Models Regression

Ünver Çiftçi

Department of Mathematics, Tekirdağ Namık Kemal University, 59030, Tekirdağ, Turkey

We use recently developed techniques in generative models, specifically normalizing flows, in regression and interpolation problems. This gives a probabilistic method which is both interpretable and efficient. Possible extensions of unsupervised learning in supervised tasks are discussed.

Density estimation | Normalizing flows| Probabilistic learning

Recent years witnessed a great advance in generative models generative adversarial networks (1), variational autoencoders (2), energy based-models (3) and normalizing flows (4). Generative models learn the data distribution, implicitly or explicitly, and generate new samples accordingly. Flow-based generative models are explicit and interpretable, so applying them to supervised task such as regression looks very promising.

Regression aims to represent a real-valued function optimally up to a measure like sum of squares loss. In this work we are formalizing regression as density estimation, and then employ flow-based methods to obtain a flexible and probabilistic framework of regression.

Our method also make it possible to generating new point estimates and interpolation of new points from the given data points. Furthermore one can find global constraint optimization of black-box function.

We know that any distribution can be transformed to any other probability distribution (under some conditions) (5), but until recently it wasn't so feasible for high-dimensional data. With the help of neural networks it is possible to generate new images(6) even videos (7).

In the next section we introduce our notation and some background on normalizing flows, then the main section we formalize flow-based regression, implementation and discussion sections outline further of applications and future directions.

Change of variables

Let $X \in \boldsymbol{R}^k$ and $Z \in \boldsymbol{R}^k$ be two random vectors and $T : Z \to X$ be a bijection or a composition of bijections. Then the following well-known formula (5):

$$P_X(\boldsymbol{x}) = P_Z(\boldsymbol{z}))|\det J_T(\boldsymbol{z})|^{-1},$$

or equivalently

$$P_X(\boldsymbol{x}) = P_Z(T^{-1}(\boldsymbol{x})|\det J_{T^{-1}}(\boldsymbol{x})|$$

is obtained, where J_T is the Jocabian determinant of T. Here $\boldsymbol{x}, \boldsymbol{z} \in \boldsymbol{R}^k$ are instances of X and Z respectively with $T(\boldsymbol{z}) = \boldsymbol{x}$. In general Z is chosen to be normally distributed and T is a flow, that is, a composition of suitably parametrized bijections. Especially, carefully designed flows parametrized with deep networks are universal approximators (8) and have been shown to scale well to real-world applications (9).

Flow-based regression

Let $X \in \boldsymbol{R}^k$ be a random vector with probability distribution $p_X(\boldsymbol{x}) = K^{-1}\exp[-f(\boldsymbol{x})]$, where $f : \boldsymbol{R}^k \to \boldsymbol{R}$ is a function and K is the normalizing constant given by $K = \int \exp[-f(\boldsymbol{x})]d\boldsymbol{x}$. We can define a bijection $T : Z \to X$ for a random vector $Z \in \boldsymbol{R}^k$ with a known distribution $P_Z(\boldsymbol{z})$ such as multivariate uniform or normal. This follows from the universality of normalizing flows (9) under mild conditions. Then from the change of variables rule

$$f(\boldsymbol{x}) = -\log P_Z(T^{-1}(\boldsymbol{x})) - \log|\det J_{T^{-1}}(\boldsymbol{x})| - \log K \qquad [1]$$

is obtained.

For a given i.i.d. data $\{\boldsymbol{x}^i\}_{i=1}^N$ we can learn $f(\boldsymbol{x})$ by parametrizing the right hand side of Equation 1. For simplicity we assume that $P_Z(\boldsymbol{z})$ is a fixed multivariate distribution but we could choose a parametrized model or even a parameterized mixture model. But T is usually taken to be a flow of bijections parametrized by a neural network. We can also train a neural network to learn K or we can use an estimator such as $K \approx \hat{K} = \sum_1^N \exp[-f(\boldsymbol{x}^i)]$ which is our choice in this work.

E-mail: uciftcinku.edu.tr

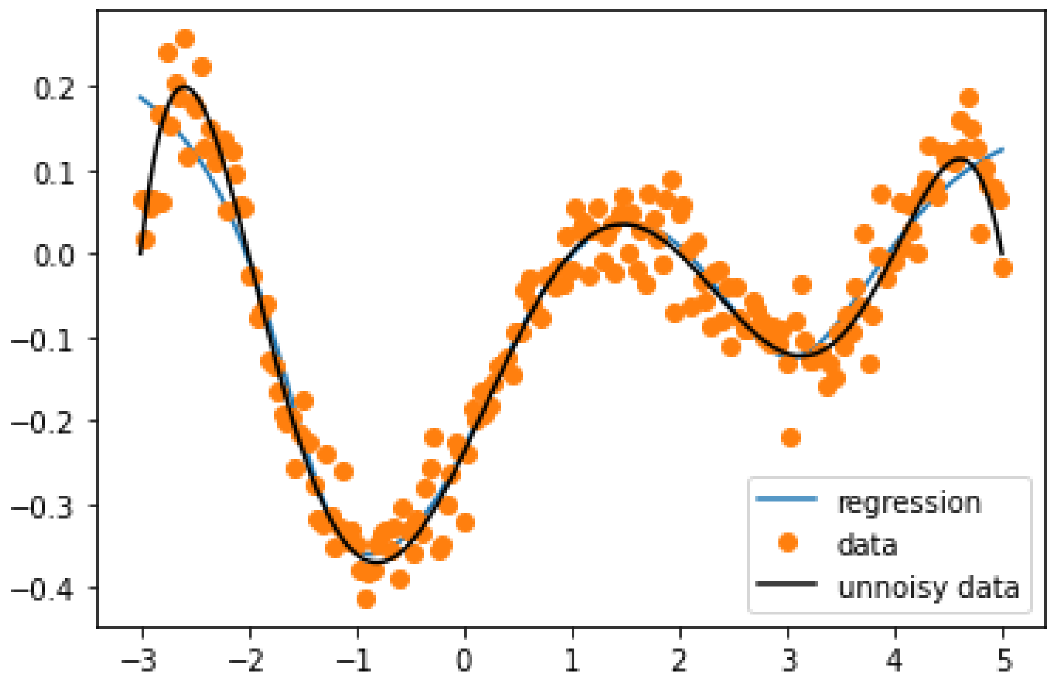

Fig. 1. Data points (yellow dots) obtained from the sixth-order polynomial curve (black line) by normally distributed noise, and the regression curve (blue line).

Now that $f(\boldsymbol{x})$ is approximated as

$$f(\boldsymbol{x}) \approx -\sum_{1}^{N}[\log P_Z(T_{\boldsymbol{\theta}}^{-1}(\boldsymbol{x}^{(i)})) + \log|\det J_{T_{\boldsymbol{\theta}}^{-1}}(\boldsymbol{x}^{(i)})|] - \log\hat{K},$$

where θ is the parameter vector to be learned. Mean squared error loss given by

$$L(\boldsymbol{\theta}) = M^{-1}\sum_{1}^{N}[f(\boldsymbol{x}^{(i)}) + \log\hat{P}_X(\boldsymbol{x}^{(i)}) + \log\hat{K}]^2,$$

where $\hat{P}_X(\boldsymbol{x}) = \log P_Z(T_{\boldsymbol{\theta}}^{-1}(\boldsymbol{x})) + \log|\det J_{T_{\boldsymbol{\theta}}^{-1}}(\boldsymbol{x})|$.

Implementation

Probabilistic programming frameworks TensorFlow Probability (TFP) (10) and Pyro (11) have API's for high level programming of flow-based models. They have functions to compute the derived distribution and sample new data, by joining bijections like hidden layers in deep learning.

We illustrate the proposed method of regression through a synthetic example. A set of 200 data points is obtained by the points of a sixth-order polynomial by adding normally distributed noise (see Figure 1). Then in TFP the base distribution is chosen to be the standard normal distribution and the flow is composed of 8 layers where each layer is an affine transfomation followed by a sinh-arcsinh transformation.

As it can be seen in Figure 1 a comparatively good regressor function (blue line) is obtained. For more layers the model learns the data better but this may cause overfitting. We can further test how certain our model is: as the base distribution is standard normal, we shift the base density in the range of ± 1 standard deviation (see Figure 2). Implementation details can be found at https://github.com/unverciftci/GMR.

Discussion

We have shown that regression as a supervised learning task can be efficiently done by using recent density estimation techniques. This leads to less human intervention and more controllable method. As the probabilistic structure is known generating new examples is just a matter of sampling, and this is a promising in interpolation. For instance finding protein structures of desired energy or generating images satisfying certain scalar constraints is possible. Although we demonstrated a one-dimensional example our method easily scales to high-dimensional data.

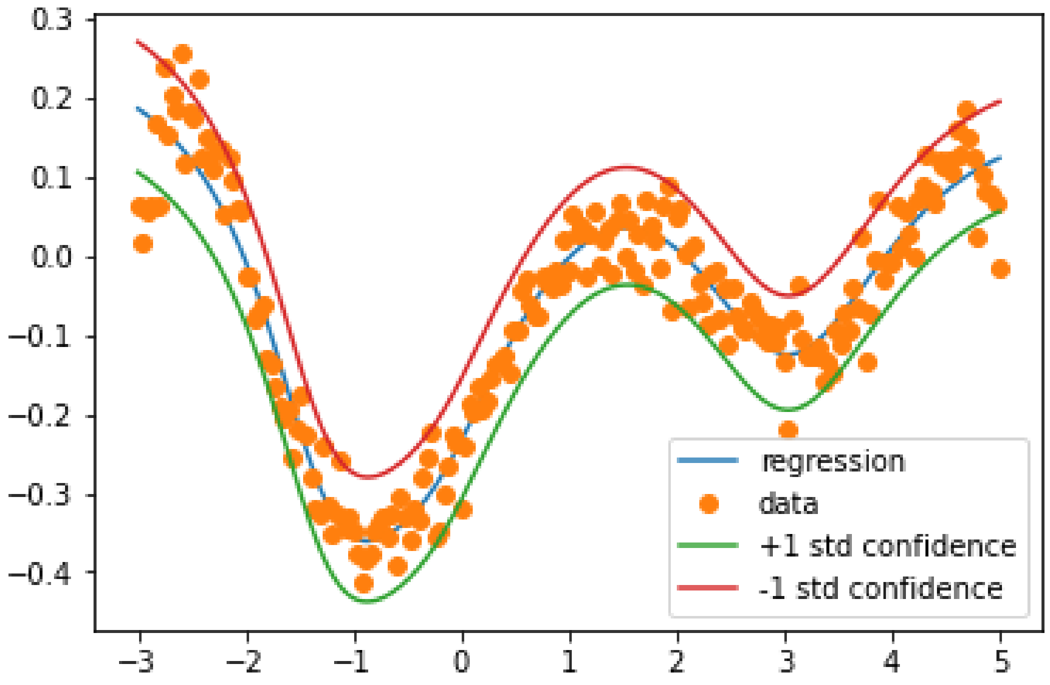

Fig. 2. Data points (yellow dots), regression curve (blue line), ± 1 standard deviation range of the regressor (red and green lines).

Our proposed method can also be extended to black-box optimization problems. Learning the average behaviour of the data by regression may help avoiding local minima. More interestingly, as the domain of the function is restricted on the data, it can be interpreted as constraint optimization which is an extremely challenging problem.

Recently there is a growing concern on employing unsupervised learning methods in supervised tasks, for instance (12), and in better understanding the data manifold, for instance, (13). We believe our point of view could serve in those directions.

Observe that for classification problems one can adopt our method because although it is a discrete probability estimation problem, suitable extensions of normalizing flows can be employed (14). Future work includes real-world applications, time-series prediction (extrapolation) and automatic learning.

1. I Goodfellow, et al., Generative adversarial nets in *Advances in neural information processing systems.* pp. 2672–2680 (2014).
2. PD Kingma, M Welling, , et al., Auto-encoding variational bayes in *Proceedings of the International Conference on Learning Representations (ICLR).* Vol. 1, (2014).
3. Y Du, I Mordatch, Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689* (2019).
4. DJ Rezende, S Mohamed, Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770* (2015).
5. VI Bogachev, *Measure theory.* (Springer Science & Business Media) Vol. 1, (2007).
6. DP Kingma, P Dhariwal, Glow: Generative flow with invertible 1x1 convolutions in *Advances in Neural Information Processing Systems.* pp. 10215–10224 (2018).
7. M Kumar, et al., Videoflow: A flow-based generative model for video. *arXiv preprint arXiv:1903.01434* (2019).
8. CW Huang, D Krueger, A Lacoste, A Courville, Neural autoregressive flows. *arXiv preprint arXiv:1804.00779* (2018).
9. G Papamakarios, E Nalisnick, DJ Rezende, S Mohamed, B Lakshminarayanan, Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762* (2019).
10. JV Dillon, et al., Tensorflow distributions. *arXiv preprint arXiv:1711.10604* (2017).
11. E Bingham, et al., Pyro: Deep universal probabilistic programming. *The J. Mach. Learn. Res.* **20**, 973–978 (2019).
12. W Grathwohl, et al., Your classifier is secretly an energy based model and you should treat it like one in *International Conference on Learning Representations.* (2019).
13. J Brehmer, K Cranmer, Flows for simultaneous manifold learning and density estimation. *arXiv preprint arXiv:2003.13913* (2020).
14. L Dinh, J Sohl-Dickstein, R Pascanu, H Larochelle, A rad approach to deep mixture models. *arXiv preprint arXiv:1903.07714* (2019).