Gradient Boosted Normalizing Flows

Robert Giaquinto and Arindam Banerjee

University of Minnesota, Twin Cities

December 7, 2020

Normalizing Flows

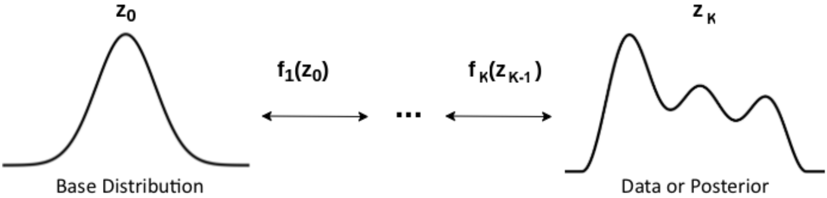


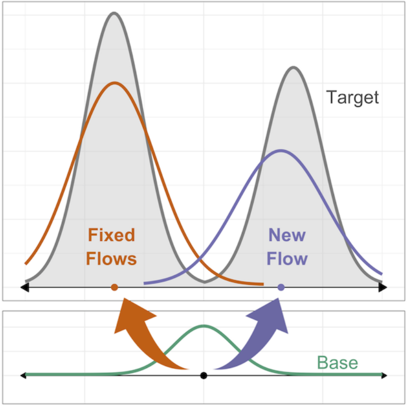
Figure: Normalizing flows construct flexible distributions via smooth, invertible mappings.

- Exact likelihoods, data generation.
- **Recent trend:** deeper, more complex transformations.

Tabak and Vanden-Eijnden 2010; Tabak and Turner 2013; Rezende and Mohamed 2015

Gradient Boosted Normalizing Flows

- A *wider* alternative
- Iteratively add new flow components
- New component fit to residuals of previous fixed components
- Density estimation
- Variational inference



Related: Rosset and Segal 2002; Guo et al. 2016; Grover and Ermon 2018; Dinh et al. 2019; Cornish et al. 2020

Density Estimation with Multiplicative GBNF

- Weighted combination of fixed and new components.

$$\text{Loss: } \mathcal{F} = -\frac{1}{N}\sum_{i=1}^{N}\left[\left(\log(G_K^{(c-1)}(\mathbf{x}_i)) + \rho_c \log(g_K^{(c)}(\mathbf{x}_i))\right) - \log \Gamma_{(c)}\right]$$

  - $\Gamma_{(c)}$ partition function
  - See paper for *additive* mixture formulation

Density Estimation with Multiplicative GBNF

- Weighted combination of fixed and new components.

$$\text{Loss: } \mathcal{F} = -\frac{1}{N}\sum_{i=1}^{N}\left[\left(\log(G_K^{(c-1)}(\mathbf{x}_i)) + \rho_c \log(g_K^{(c)}(\mathbf{x}_i))\right) - \log \Gamma_{(c)}\right]$$

  - $\Gamma_{(c)}$ partition function
  - See paper for *additive* mixture formulation

- Training GBNF:

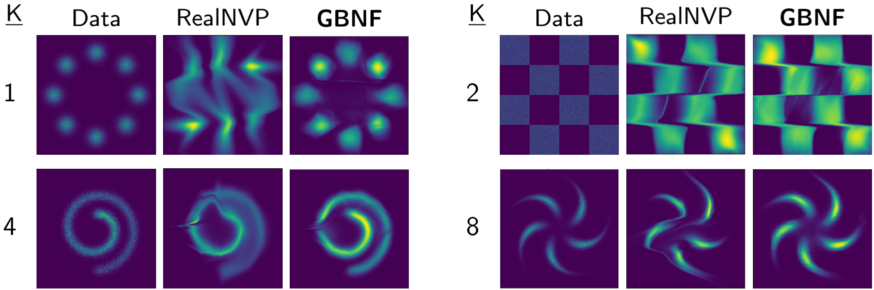
  Component c=1 : Fit with traditional objective — no boosting!

  Component $c > 1$: Have fixed components $G_K^{(c-1)}$, and new component $g_K^{(c)}$

  1. Train $g_K^{(c)}$ via Frank-Wolfe linear approximation
  2. Optimize weight $\rho_c \in [0, 1]$

Advantages of Gradient Boosted Normalizing Flows

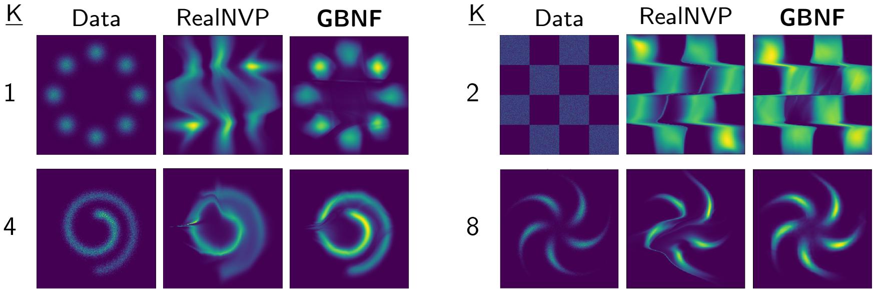
- Compliments existing flow models[1]



[1] Note: Variational inference requires *analytically* invertible flows (see paper).

Advantages of Gradient Boosted Normalizing Flows

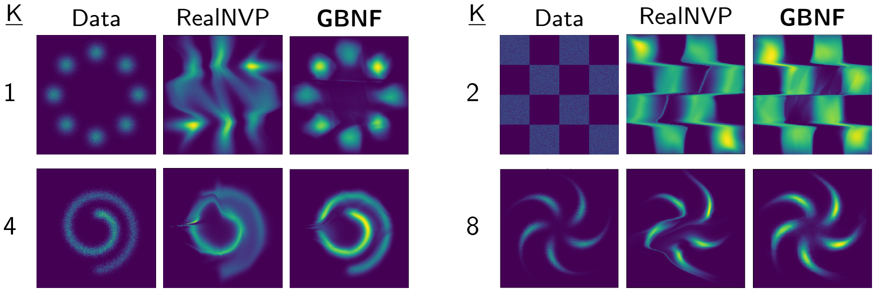
- Compliments existing flow models[1]
- Resembles mixture
  - Easier! Just optimize $g_K^{(c)}$



[1]Note: Variational inference requires *analytically* invertible flows (see paper).

Advantages of Gradient Boosted Normalizing Flows

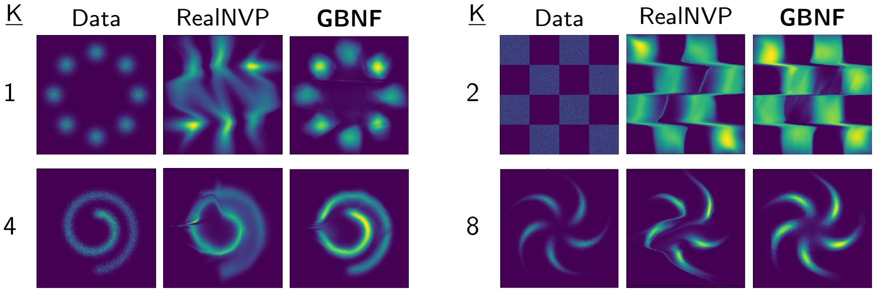
- Compliments existing flow models[1]
- Resembles mixture
  - Easier! Just optimize $g_K^{(c)}$
- Exchange flexibility for training cycles



[1]Note: Variational inference requires *analytically* invertible flows (see paper).

Advantages of Gradient Boosted Normalizing Flows

- Compliments existing flow models[1]
- Resembles mixture
  - Easier! Just optimize $g_K^{(c)}$
- Exchange flexibility for training cycles
- Prediction/sampling in parallel



[1] Note: Variational inference requires *analytically* invertible flows (see paper).

See Our Paper for More!

- Training components
- Analysis of objectives
- Unique challenges (“Decoder Shock”)
- Experiments

Thank you!

R. Cornish, A. L. Caterini, G. Deligiannidis, and A. Doucet. Relaxing Bijectivity Constraints with Continuously Indexed Normalising Flows. *arXiv:1909.13833 [cs, stat]*, Feb. 2020.

L. Dinh, J. Sohl-Dickstein, R. Pascanu, and H. Larochelle. A RAD approach to deep mixture models. *arXiv:1903.07714 [cs, stat]*, Mar. 2019.

A. Grover and S. Ermon. Boosted Generative Models. In *AAAI*, 2018.

F. Guo, X. Wang, K. Fan, T. Broderick, and D. B. Dunson. Boosting Variational Inference. In *Advances in Neural Information Processing Systems*, Barcelona, Spain, 2016.

D. J. Rezende and S. Mohamed. Variational Inference with Normalizing Flows. In *International Conference on Machine Learning*, volume 37, pages 1530–1538, Lille, France, 2015. PMLR.

S. Rosset and E. Segal. Boosting Density Estimation. In *Advances in Neural Information Processing Systems*, page 8, 2002.

E. G. Tabak and C. V. Turner. A Family of Nonparametric Density Estimation Algorithms. *Communications on Pure and Applied Mathematics*, 66(2): 145–164, Feb. 2013. ISSN 00103640. doi: 10.1002/cpa.21423.

E. G. Tabak and E. Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010. ISSN 15396746, 19450796. doi: 10.4310/CMS.2010.v8.n1.a11.

Optimizing a New (Multiplicative) Boosting Component

- Fit $g_K^{(c)}$ based on *functional* gradient descent, yields:

$$g_K^{(c)} = \underset{g_K \in \mathcal{G}_K}{\arg\max} \; \mathbb{E}_{p^*} [\log g_K(\mathbf{x})] - \log \mathbb{E}_{G_K^{(c-1)}} [g_K(\mathbf{x})]$$

Optimizing a New (Multiplicative) Boosting Component

- Fit $g_K^{(c)}$ based on *functional* gradient descent, yields:

$$g_K^{(c)} = \underset{g_K \in \mathcal{G}_K}{\arg\max} \; \mathbb{E}_{p^*} \left[\log g_K(\mathbf{x})\right] - \log \mathbb{E}_{G_K^{(c-1)}} \left[g_K(\mathbf{x})\right]$$

- Solution given by:

$$g_K^{(c)}(\mathbf{x}) = \frac{p^*(\mathbf{x})}{G_K^{(c-1)}(\mathbf{x})}$$

Optimizing a New (Multiplicative) Boosting Component

- Fit $g_K^{(c)}$ based on *functional* gradient descent, yields:

$$g_K^{(c)} = \underset{g_K \in \mathcal{G}_K}{\arg\max} \ \mathbb{E}_{p^*}\left[\log g_K(\mathbf{x})\right] - \log \mathbb{E}_{G_K^{(c-1)}}\left[g_K(\mathbf{x})\right]$$

- Solution given by:

$$g_K^{(c)}(\mathbf{x}) = \frac{p^*(\mathbf{x})}{\color{red}{G_K^{(c-1)}(\mathbf{x})}}$$

- How will new $g_K^{(c)}$ behave?

Optimizing a New (Multiplicative) Boosting Component

- Fit $g_K^{(c)}$ based on *functional* gradient descent, yields:

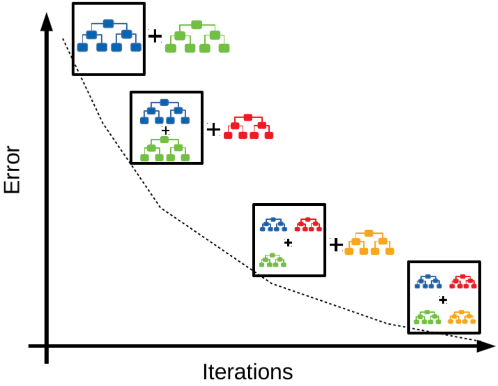
$$g_K^{(c)} = \underset{g_K \in \mathcal{G}_K}{\arg\max}\ \mathbb{E}_{p^*}\left[\log g_K(\mathbf{x})\right] - \log \mathbb{E}_{G_K^{(c-1)}}\left[g_K(\mathbf{x})\right]$$

- Solution given by:

$$g_K^{(c)}(\mathbf{x}) = \frac{p^*(\mathbf{x})}{G_K^{(c-1)}(\mathbf{x})}$$

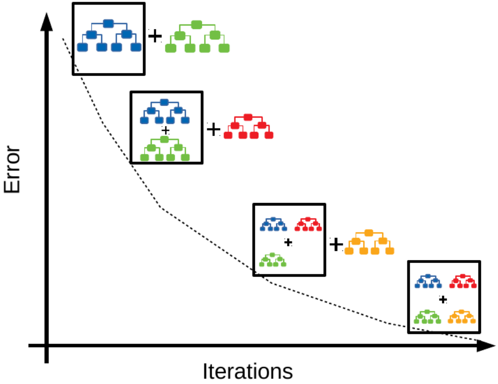
- How will new $g_K^{(c)}$ behave?
  - Large $G_K^{(c-1)} \implies$ ignore, already covered
  - Small $G_K^{(c-1)} \implies$ attractive!

Decoder Shock



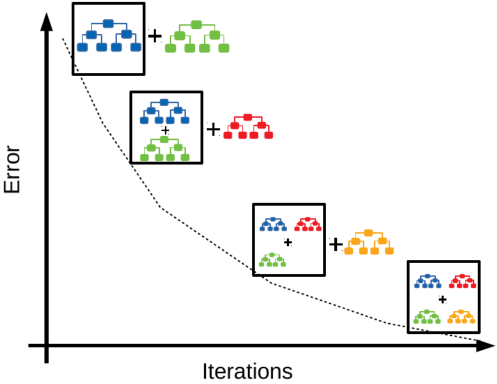
- Gradient boosting flows unlike decision trees

Decoder Shock



- Gradient boosting flows unlike decision trees
- Decoder shared by all components

Decoder Shock



- Gradient boosting flows unlike decision trees
- Decoder shared by all components
- What happens when we begin training a new component?

Decoder Shock

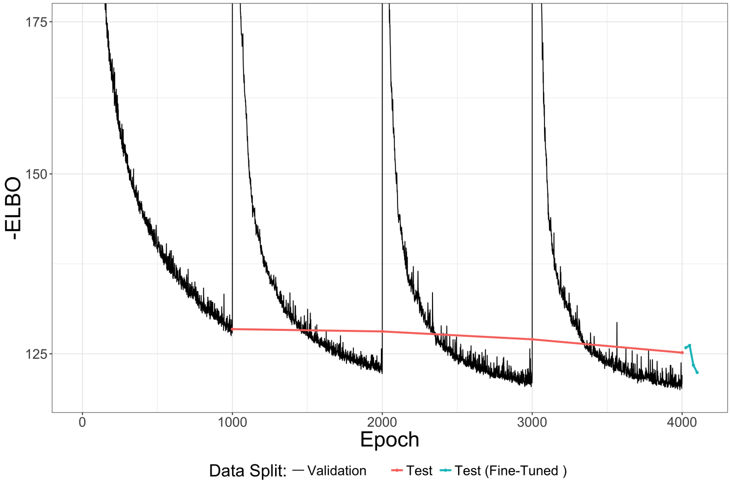


Figure: Loss on the test set decreases steadily as we add new components. The validation loss, however, jumps when a new component is introduced due to a sudden change in the distribution of samples passed to the decoder (aka "decoder shock").

Reasons for Decoder Shock

New component = sudden shift in distribution of samples, why?

1. Samples coming from *new* component
2. Objective's regularization $KL(q(\mathbf{z} \mid \mathbf{x} \,||\, p(\mathbf{z}))$ annealed from 0 to 1
   $\implies$ No regularization (temporally)
   $\implies$ Model is free to find very flexible posterior

Solution:

Reasons for Decoder Shock

New component = sudden shift in distribution of samples, why?

1. Samples coming from *new* component
2. Objective's regularization $KL(q(\mathbf{z} \mid \mathbf{x} \,||\, p(\mathbf{z}))$ annealed from 0 to 1
   $\Longrightarrow$ No regularization (temporally)
   $\Longrightarrow$ Model is free to find very flexible posterior

Solution:

- Blend in samples from fixed components
- Helps "remember" previous components
- $G_{1:K}^{(c-1)}$ still fixed