Preserving Properties of Neural Networks by Perturbative Updates

Andreas Krämer, Jonas Köhler, Frank Noé

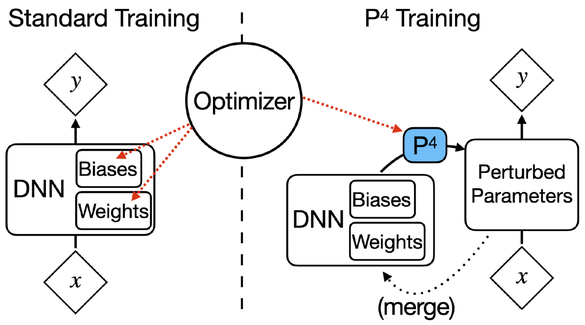
Freie Universität Berlin

AI4Science Group
Department of Mathematics and Computer Science

arXiv preprint arXiv:2010.07033

Introduction

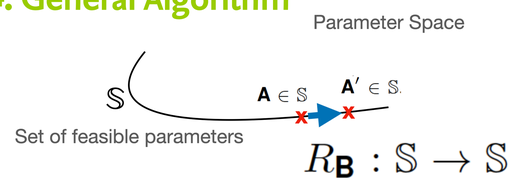
- Machine learning (especially in the physical sciences) often requires networks/layers that obey certain properties
  - Invariances, Equivariances
  - (Lipschitz) Regularity
  - Invertibility, Orthogonality, …
- Retaining such properties during training can be challenging
- Existing approaches: enforce properties by network design; on-manifold optimization with Riemannian gradient descent; reparameterization; dynamic trivializations



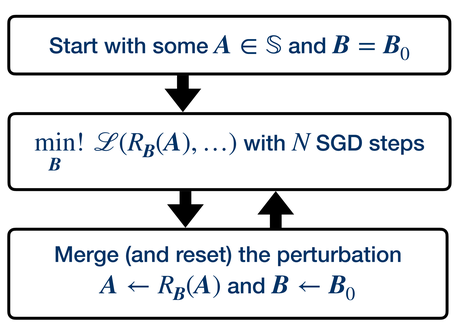
- Idea: train neural networks through *property-preserving parameter perturbations* (P4)
  1. Freeze the network parameters
  2. Optimize a perturbation to these parameters that preserves the desired properties
  3. In regular intervals: merge the perturbation into the frozen parameters
- Here we train invertible linear layers through rank-one perturbations (P4Inv)

Method

P4: General Algorithm



- Define a P4, i.e., a parameterized perturbation $R_{\boldsymbol{B}} : \mathbb{S} \to \mathbb{S}$ that includes the identity (i.e., there exists a $\boldsymbol{B}_0$ with $R_{\boldsymbol{B}_0} = id_{\mathbb{S}}$)
- Replace the loss function $\mathscr{L}(\boldsymbol{A}, \dots)$ by the reparameterization $\mathscr{L}(R_{\boldsymbol{B}}(\boldsymbol{A}), \dots)$



P4Inv: Preserving Invertibility

- Keep track of inverses and determinants
- Can be used in *normalizing flows*, reversible generative models that require tractable computation of inverse and Jacobian determinant
- $\mathbb{S}$:= set of invertible matrices with known inverses and determinants
- $R_{\boldsymbol{B}}$:= rank-one perturbation with $\boldsymbol{B} = (\boldsymbol{u}, \boldsymbol{v})$

$$R_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{A}) = \boldsymbol{A} + \boldsymbol{u}\boldsymbol{v}^T$$

$$(A + uv^T)^{-1} = A^{-1} - \frac{1}{1 + v^T A^{-1} u} A^{-1} u v^T A^{-1}$$

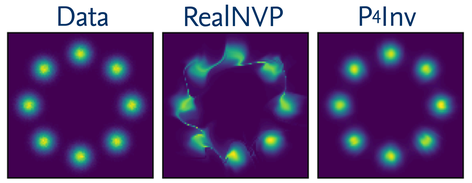
$$\det(\boldsymbol{A} + uv^T) = (1 + v^T \boldsymbol{A}^{-1} u)\det(\boldsymbol{A})$$

- $\boldsymbol{B}_0$:= reset to $id_{\mathbb{S}}$ via $\boldsymbol{v} := 0$, $u_i \sim \mathcal{N}(0,1)$
- Merging step with numerical stabilization
  - Skip merge if the perturbed determinant signals ill conditioning of the update
  - Optionally augment the loss function by a penalty term that prevents vanishing determinants
  - Infrequently run one iteration of an iterative matrix inversion to remove numerical errors in the inverse

Experiments

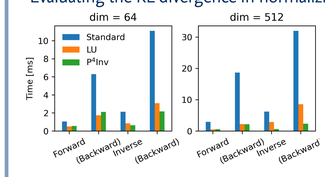
Density Estimation

- Alternate P4Inv layers with a smooth, bijective activation function
- Better mixing and mode separation than RealNVP coupling layers



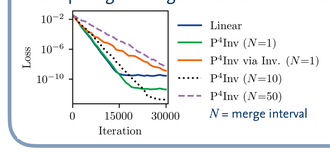
Efficiency

Evaluating the KL divergence in normalizing flows



Highly efficient inverse pass compared to LU decomposed layers and standard inversion.

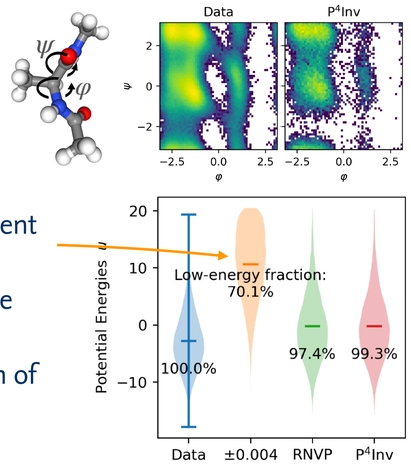
Comparing convergence to standard linear layers



P4Inv training with SGD is almost as fast as standard training on linear test problems.

Boltzmann Generators [2]

- Train a generative model for 66-dimensional alanine dipeptide conformations [3] through a combination of forward and inverse KL divergence
- Replace permutation layers in a RealNVP flow by P4Inv
- Highly sensitive potential energy: 0.004 nm displacement of the data increases the average energy by 13 $k_B T$
- The P4Inv-augmented RealNVP flow generated accurate samples, average potential energy shift: 2.6 $k_B T$
- The generated samples reproduce the joint distribution of backbone torsions well
- This result indicates sound inverses and determinants



Conclusions

- P4Inv layers can approximate arbitrary invertible matrices → expressive normalizing flows with a cheap inverse pass
- P4 training preserves properties of individual layers or the whole network during training; applicable whenever we can define a suitable property-preserving perturbation

References

[1] Andreas Krämer, Jonas Köhler, Frank Noé. Training Invertible Linear Layers through Rank-One Perturbations, arXiv preprint arXiv:2010.07033, 2020.

[2] Frank Noé, Simon Olsson, Jonas Köhler, Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. Science, 365(6457):eaaw1147, 2019.

[3] Manuel Dibak, Leon Klein, and Frank Noé. Temperature-steerable flows. Machine Learning and the Physical Sciences, Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS) , arXiv preprint arXiv:2012.00429, 2020.