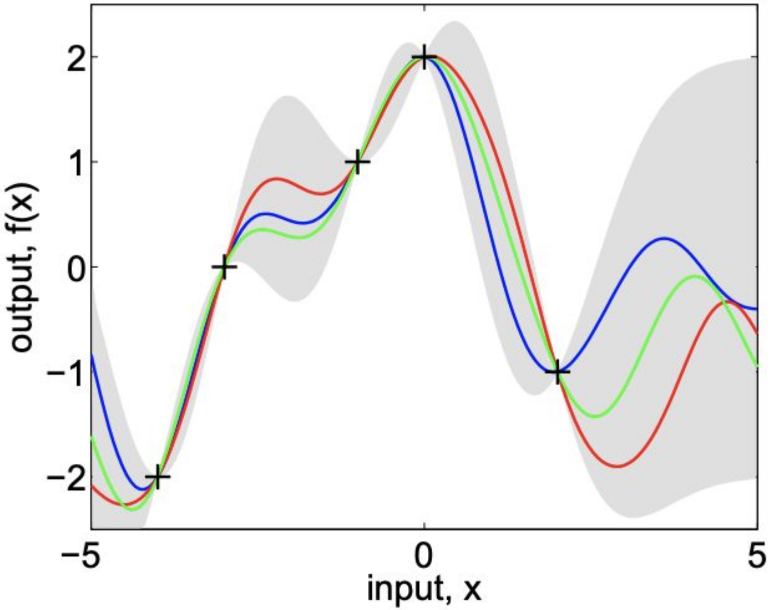
Bootstrapping Neural Process

Juho Lee*, Yoonho Lee*, Jungtaek Kim,
Eunho Yang, Sung Ju Hwang, Yee Whye Teh
Published at NeurIPS 2020

Graduate school of AI, KAIST

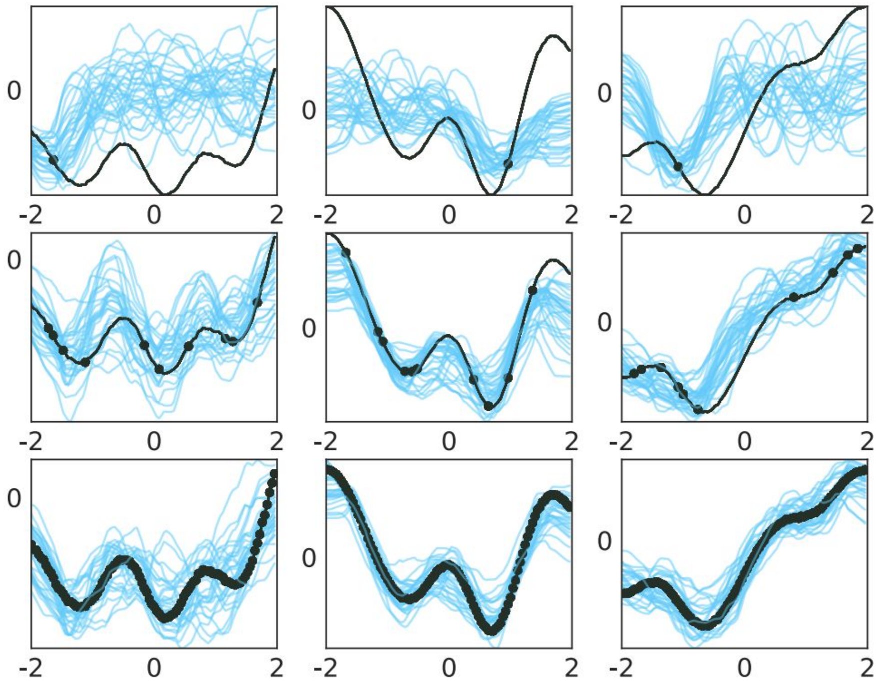
Gaussian process regression



Img taken from [1]

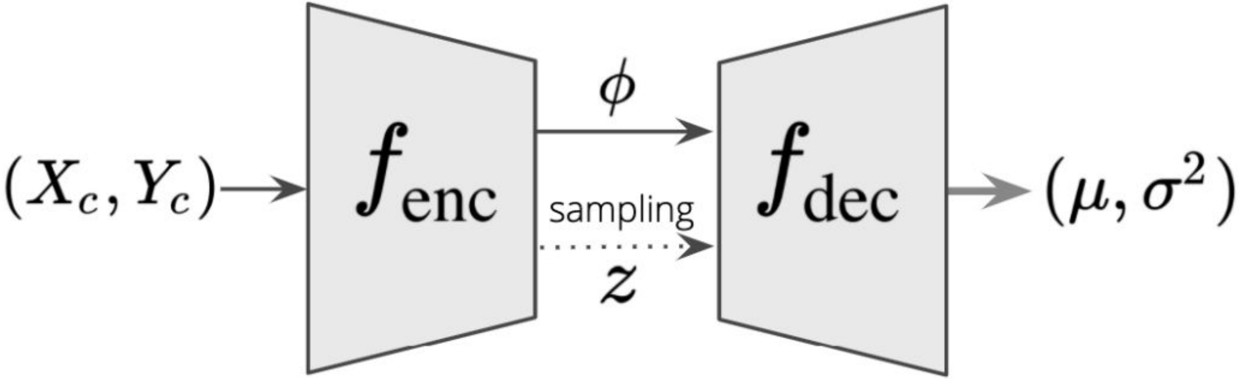
- A prior over functions (stochastic processes).
- Assuming smoothness of functions formulated with Gaussian distributions.
- Closed-form predictive distributions.
- Inefficient for large-scale data.
- Not suitable for high-dimensional data.
- May not work well for data with non-Gaussian errors.

Neural processes (NPs) [2]



- An implicitly defined prior over functions.
- A neural network version of stochastic processes.
- A data-driven way of learning priors over functions.
- Fast inference once trained.
- May describe high-dimensional and non-Gaussian data better.

Structure of NPs



$$\phi = f_{\text{denc}}(X_c, Y_c), \quad (\eta, \rho) = f_{\text{lenc}}(X_c, Y_c), \quad q(z|X_c, Y_c) = \mathcal{N}(z; \eta, \rho^2)$$
$$(\mu_i, \sigma_i) = f_{\text{dec}}(\phi, z, x_i), \quad p(y_i|x_i, z, \phi) = \mathcal{N}(y_i|\mu_i, \sigma_i^2),$$

Training NPs

- Collect the training datasets (not a single data, collect multiple training datasets from a task distribution).
- Construct a neural process model.
- Maximize the lower-bound on the expected likelihood.

$$\log p(Y|X, Y_c) \geq \sum_{i=1}^{n} \mathbb{E}_{q(z|X,Y)} \left[\log \frac{p(y_i|x_i, z, \phi) p(z|X_c, Y_c)}{q(z|X, Y)} \right].$$

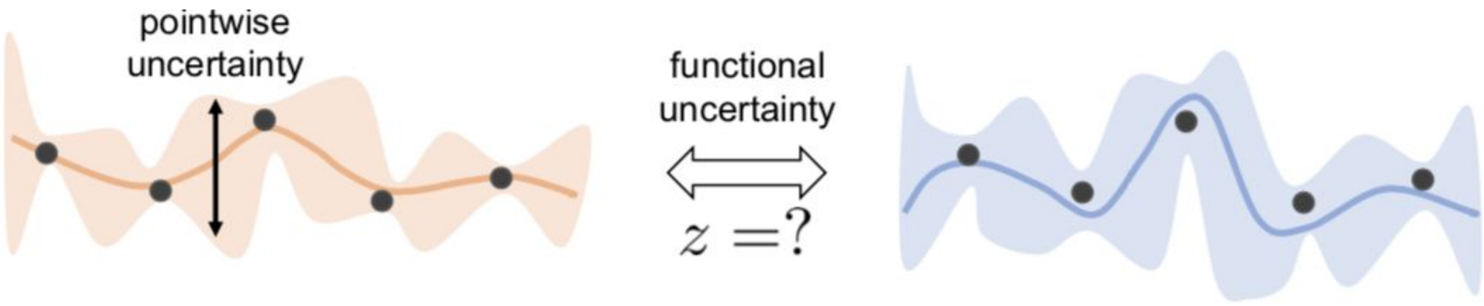
Uncertainties in NPs

- Aleatoric uncertainty (pointwise uncertainty, measurement uncertainty)

$$p(y_i|x_i, z, \phi) = \mathcal{N}(y_i|\mu_i, \sigma_i^2)$$

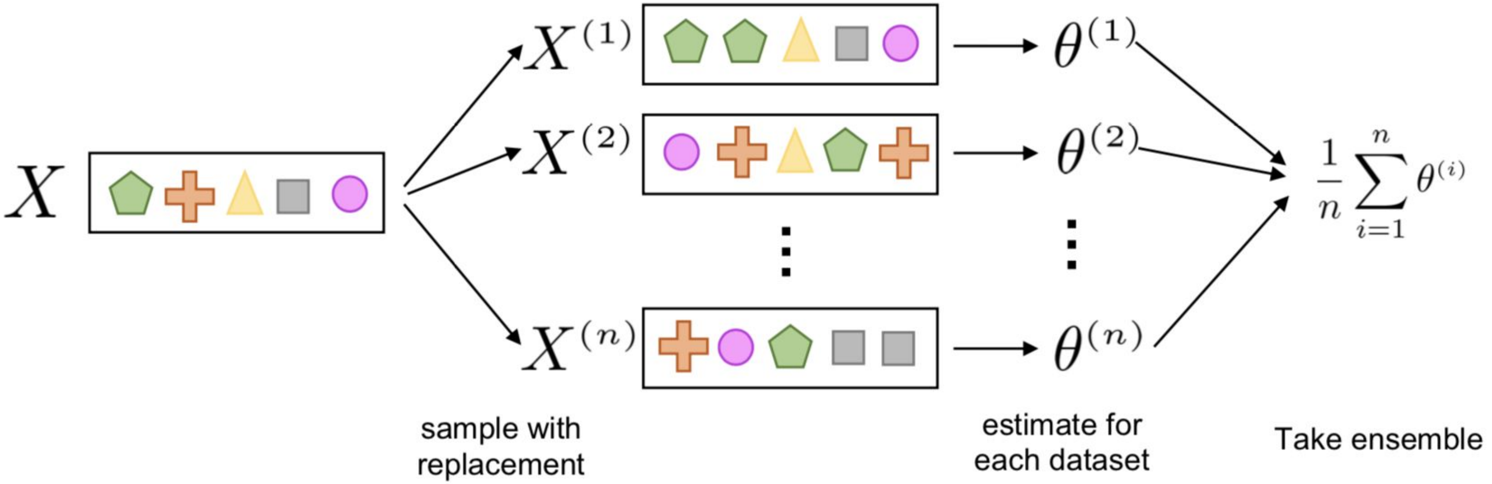
- Epistemic uncertainty (**functional** uncertainty, model uncertainty)

$$(\eta, \rho) = f_{\text{lenc}}(X_c, Y_c), \quad q(z|X_c, Y_c) = \mathcal{N}(z; \eta, \rho^2)$$



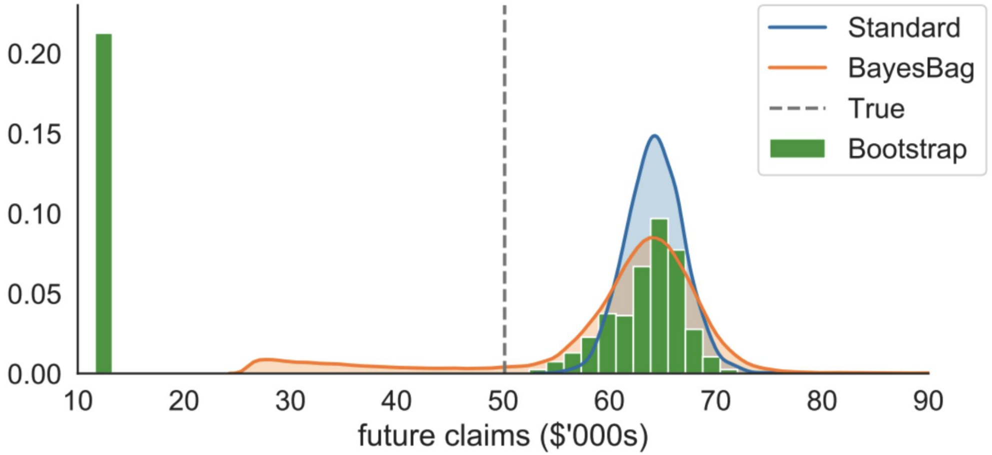
Bootstrap and Bagging

- Bootstrap [3]: use resampling to simulate distribution of parameters.
- Bagging [4]: ensembling estimates from multiple bootstrap datasets.



BayesBag [5]

- Bagging posteriors instead of point estimates.
- Requires less number of bootstrap samples.
- Robust under model-data mismatch [5].

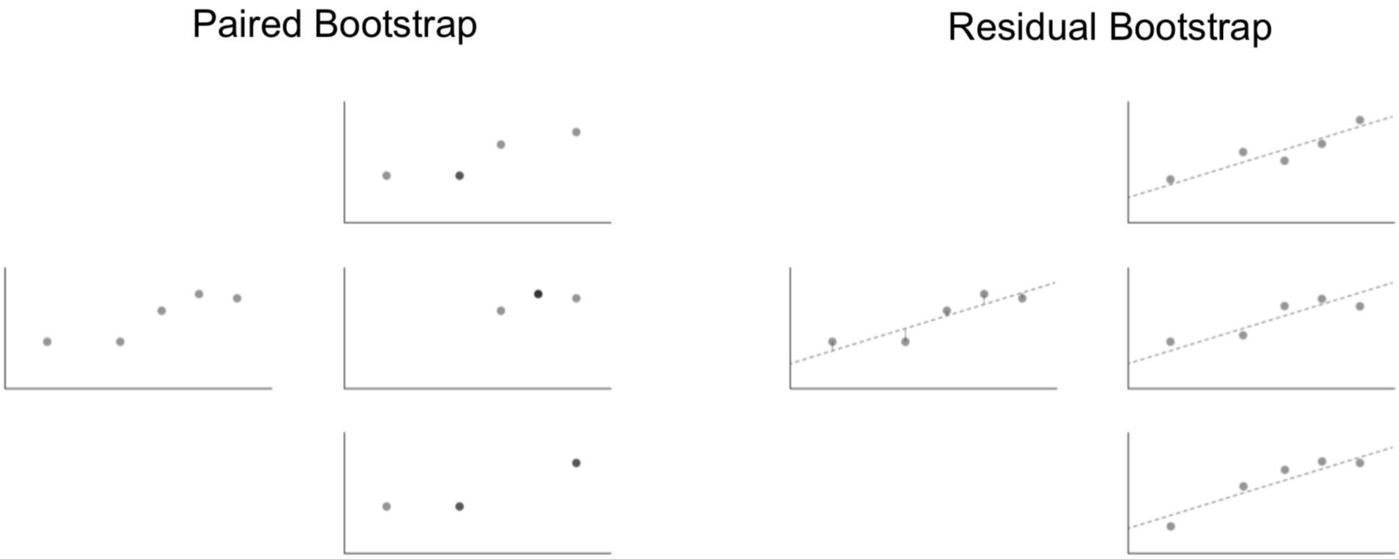


Bootstrapping neural processes (BNP) - motivation

- BNP in a nutshell: replace the latent variable with bootstrapping to induce functional uncertainty.
- More natural way to model functional uncertainty
  - Modeling functional uncertainty with a gaussian latent variable is unnatural, and may act as a bottleneck.
  - We want to minimize the modeling assumption and let the data describe.
  - Bootstrap and bagging may be a better way to model uncertainty in this case.
- Model-data mismatch
  - Recall that a neural process requires data sampled from a task distribution. What happens if it encounters data not from the task distribution seen during the training?
  - BayesBag is proven to be robust under such scenario. Can neural process benefit from this property?

BNP - using residual bootstrap

- We use residual bootstrap instead of vanilla bootstrap.



BNP - using residual bootstrap

1. Compute the initial prediction using the model.

$$\hat{\phi}^{(j)} = f_{\text{enc}}(\hat{X}^{(j)}, \hat{Y}^{(j)}), \quad (\hat{\mu}_i^{(j)}, \hat{\sigma}_i^{(j)}) = f_{\text{dec}}(x_i, \hat{\phi}^{(j)}) \text{ for } i \in c.$$

2. Compute the residuals and resample them.

$$\varepsilon_i^{(j)} = \frac{y_i - \hat{\mu}_i^{(j)}}{\hat{\sigma}_i^{(j)}} \text{ for } i \in c, \;\; \mathcal{E}^{(j)} = \{\varepsilon_i^{(j)}\}_{i=1}^{c}, \;\; \tilde{\varepsilon}_1^{(j)}, \ldots, \tilde{\varepsilon}_{|c|}^{(j)} \overset{\text{s.w.r.}}{\sim} \mathcal{E}^{(j)}.$$

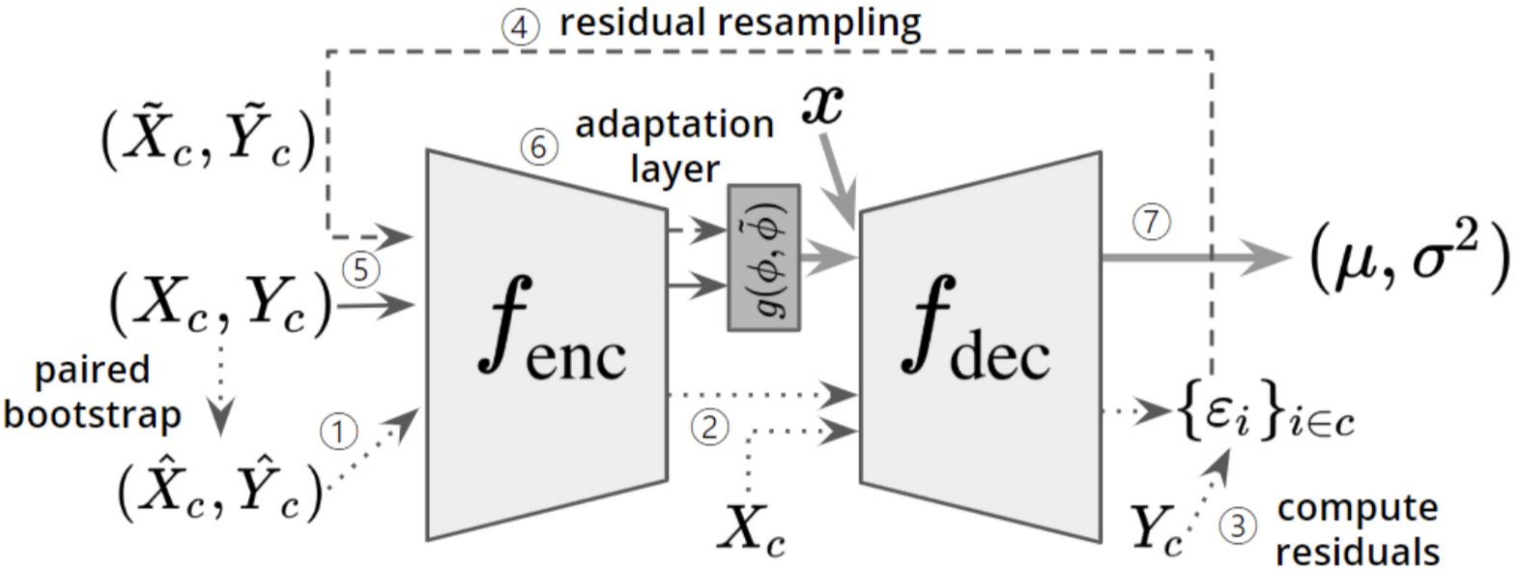
3. Construct bootstrap datasets.

$$\tilde{x}_i^{(j)} = x_i, \;\; \tilde{y}_i^{(j)} = \hat{\mu}_i^{(j)} + \hat{\sigma}_i^{(j)} \tilde{\varepsilon}_i^{(j)} \text{ for } i \in c,$$
$$(\tilde{X}_c^{(j)}, \tilde{Y}_c^{(j)}) := \{(\tilde{x}_i^{(j)}, \tilde{y}_i^{(j)})\}_{i \in c} \text{ for } j = 1, \ldots, k.$$

BNP - naive application does not work

- We can simply take a trained neural network and apply residual bootstrap + bagging.
- Unfortunately, this works terribly, even worse than the original NP.
  - The approximate posteriors computed from the NP are not perfectly accurate.
  - The bootstrap data generated from residual bootstrap are different from the training distribution, so they act as OOD data.
- We need to train NPs with bootstrap procedure!

BNP - architecture



BNP - a forward pass

- A BNP passes data into the model twice in a single forward pass.
- In the first forward pass, the data passes through the encoder-decoder to construct an initial prediction and compute the residuals.
- Once bootstrap samples are constructed, they are passed through the encoder-decoder again to compute predictions.
- The final prediction is given as a bagged posterior from multiple bootstrap samples.

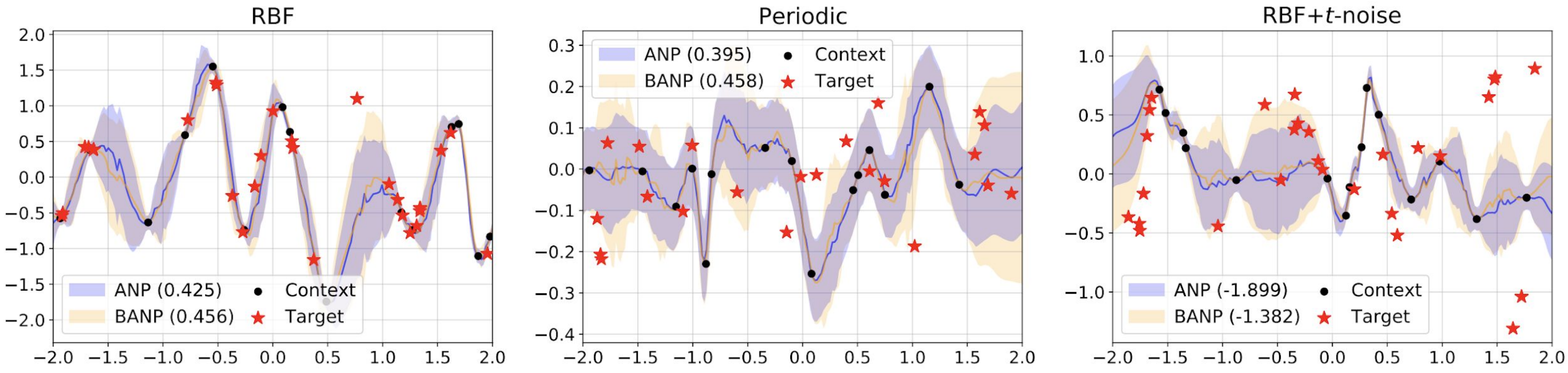
BNP - remark

- Every computation can be parallelized - inference for multiple bootstrap datasets can be done by packing them into a tensor.
- Why do we expect BNP to be robust under model-data mismatch?
  - It is an amortized version of BayesBag, so expected to enjoy the robustness of it.
  - When test data is different from training, then we would have large residuals in the initial prediction, and this encourage the model to be conversative in the final prediction.

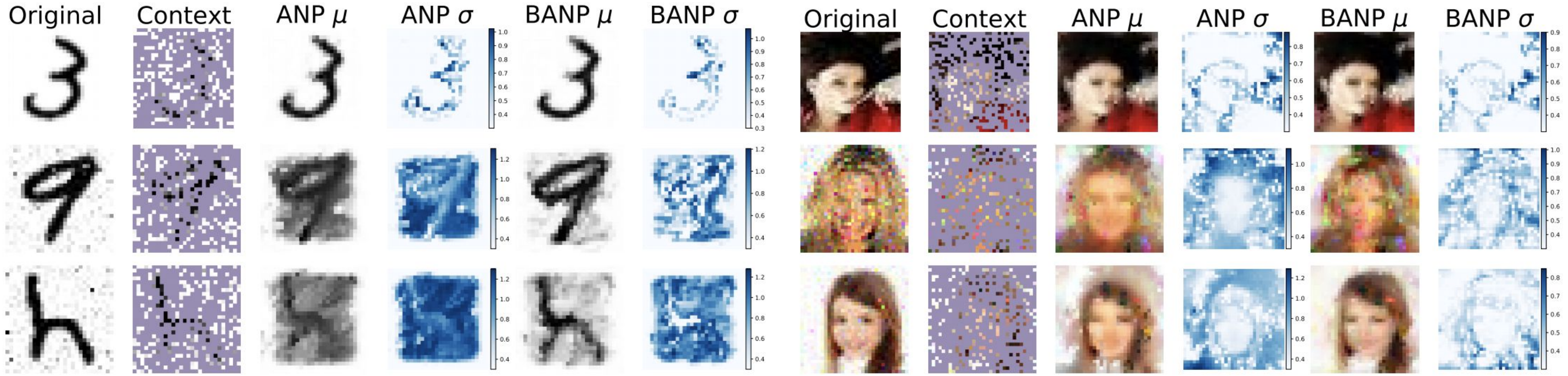
BNP - 1d regression experiments

	RBF		Matérn 5/2		Periodic		t-noise	
	context	target	context	target	context	target	context	target
CNP	0.972$\pm$0.008	0.448$\pm$0.006	0.846$\pm$0.009	0.206$\pm$0.006	-0.163$\pm$0.008	-1.747$\pm$0.023	0.363$\pm$0.147	-1.528$\pm$0.068
NP	0.902$\pm$0.009	0.420$\pm$0.008	0.774$\pm$0.012	0.204$\pm$0.010	-0.181$\pm$0.010	-1.338$\pm$0.025	0.442$\pm$0.016	-0.792$\pm$0.048
CNP+DE	0.995	0.521	0.878	**0.313**	**-0.098**	-1.384	0.534	-1.129
BNP	**1.013**$\pm$0.007	**0.526**$\pm$0.005	**0.890**$\pm$0.009	**0.317**$\pm$0.006	-0.112$\pm$0.007	**-1.082**$\pm$0.011	**0.553**$\pm$0.009	**-0.630**$\pm$0.014
CANP	**1.379**$\pm$0.000	0.838$\pm$0.001	1.376$\pm$0.000	0.652$\pm$0.001	0.476$\pm$0.043	-5.896$\pm$0.134	1.104$\pm$0.009	-2.243$\pm$0.031
ANP	**1.379**$\pm$0.000	0.842$\pm$0.002	1.376$\pm$0.000	0.660$\pm$0.001	0.600$\pm$0.034	-4.357$\pm$0.182	1.125$\pm$0.003	**-1.776**$\pm$0.021
CANP+DE	1.378	0.847	1.376	0.670	**0.771**	-4.598	**1.161**	-1.991
BANP	**1.379**$\pm$0.000	**0.851**$\pm$0.002	1.376$\pm$0.000	**0.672**$\pm$0.001	0.705$\pm$0.016	**-3.275**$\pm$0.114	1.142$\pm$0.007	**-1.718**$\pm$0.055

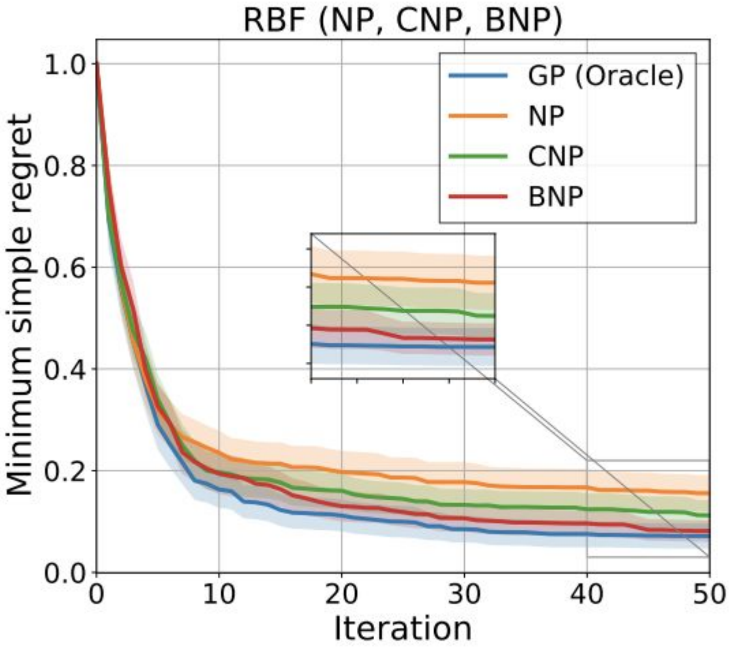
BNP - 1d regression experiments

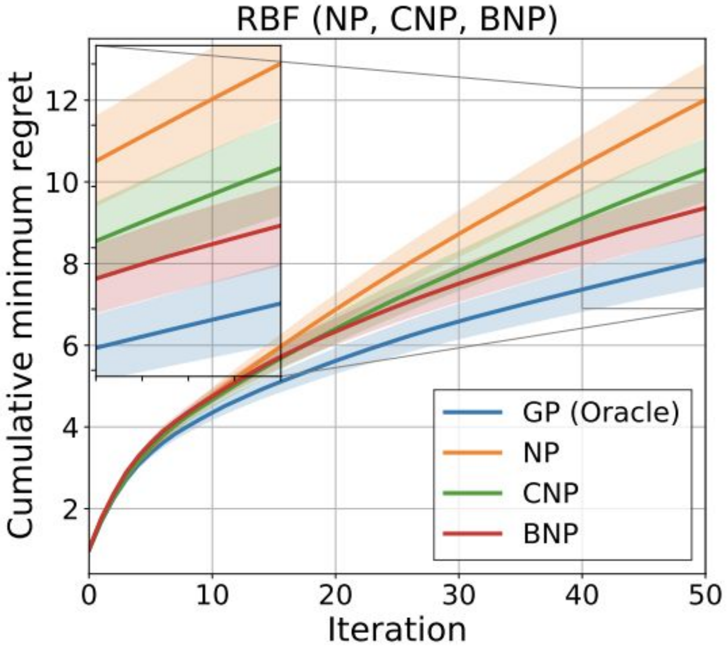


BNP - image completion experiments



BNP - Bayesian optimization



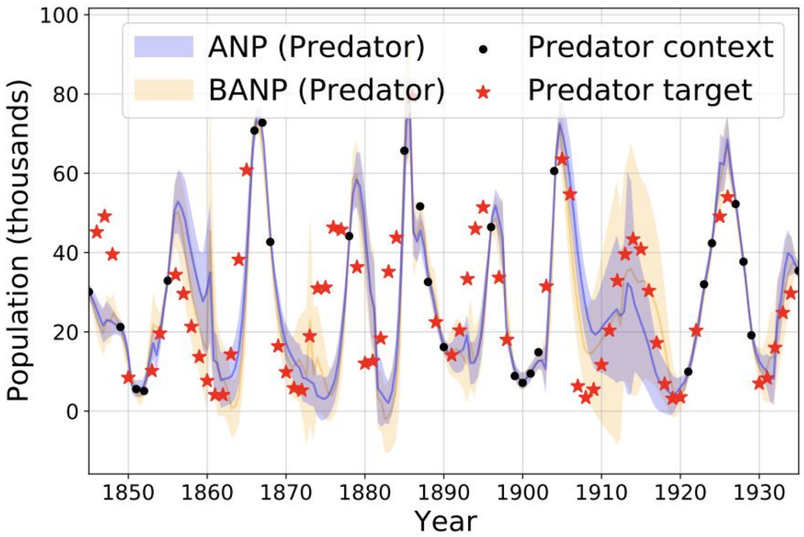


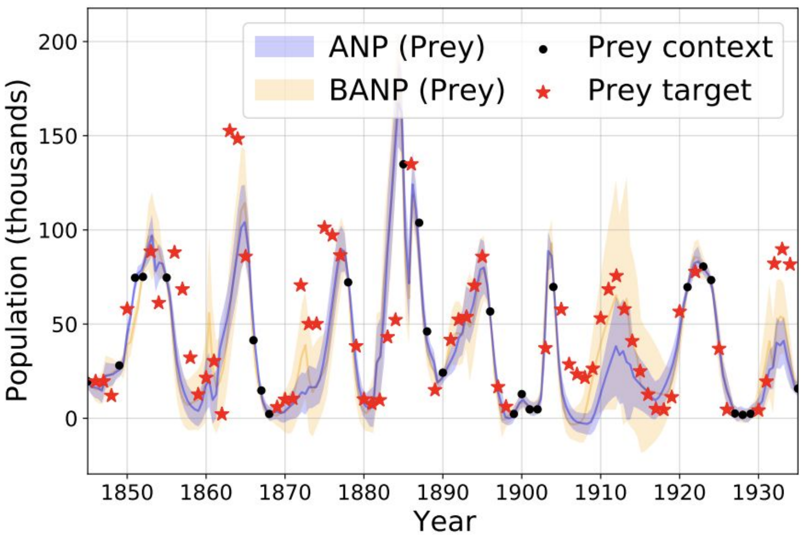
BNP - Predator-prey model

- Training models using data simulated from a differential equation model (Lotka-Volterra model), and test on real data (Hare-lynx data).
- All the models do well on simulated data, but fail on real data.

	Simulated		Real	
	context	target	context	target
CNP	0.088 ± 0.031	-0.142 ± 0.028	-2.702 ± 0.007	-3.013 ± 0.025
NP	-0.002 ± 0.039	-0.252 ± 0.036	-2.747 ± 0.019	-3.057 ± 0.020
CNP+DE	0.176	**-0.026**	-2.670	**-2.952**
BNP	**0.213** ± 0.045	**-0.011** ± 0.041	**-2.654** ± 0.005	**-2.942** ± 0.010
CANP	2.573 ± 0.014	1.819 ± 0.021	1.767 ± 0.089	-8.007 ± 0.538
ANP	2.582 ± 0.007	1.828 ± 0.007	1.720 ± 0.257	-7.809 ± 0.642
CANP+DE	**2.591**	**1.874**	**2.021**	**-5.440**
BANP	2.586 ± 0.009	1.855 ± 0.009	1.783 ± 0.156	**-5.465** ± 0.278

BNP - Predator-prey model





Conclusion

- Proposed BNP - a more natural way of incorporating model uncertainty.
- Residual bootstrap works well for regression tasks, and more robust under model-data mismatch.
- Future works
  - BNP for classifications
  - Other bootstrap strategies - wild bootstrap, parametric bootstrap, …
  - Applications to recent variants of NPs (convolutional, group-equivariant, …)

References

[1] C. E. Rasmussen and C. K. Williams, Gaussian process in machine learning, MIT Press, 2006.

[2] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. W. Teh. Neural processes. ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models, 2018.

[3] B. Efron. Bootstrap methods: another look at the jackknife. Annals of Statistics, 7(1):1–26, 1979.

[4] L. Breiman. Bagging predictors. Machine Learning, 24(2):123–140, 1996.