UNIVERSITY OF COPENHAGEN
DEPARTMENT OF COMPUTER SCIENCE

EinStein VI: General and Integrated Stein Variational Inference in NumPyro

Ahmad Salim Al-Sibahi (*UCPH*), Ola Rønning (*UCPH*), Christophe Ley (*UGent*), Thomas Hamelryck (*UCPH*)

Introduction and User Interface

Model

```
def model(X, y=None):
  Wl = np.zeros((4, 3))
  Ws = np.ones((4, 3))

  bl = np.zeros(3)
  bs = np.ones(3)

  W = sample('Wl', Normal(Wl, Ws))
  b = sample('bl', Normal(bl, bs))
  probs = softmax(X @ W + b)
  with plate('data', X.shape[0]):
    sample('y', Categorical(probs),
           obs=y)
```

Inference Program

```
def guide(X, y=None):
  Wl = param('Wl', np.zeros((4, 3)))
  Ws = param('Ws', np.ones((4, 3)),
             constraint=positive)
  bl = param('bl', np.zeros(3))
  bs = param('bs', np.ones(3),
             constraint=positive)
  W = sample('W', Normal(Wl, Ws))
  b = sample('b', Normal(bl, bs))
```

Training

```
data = load_iris()
X_train, X_test, y_train, y_test = \
    train_test_split(data.data, data.target)

einstein = EinStein(model, WrappedGuide(guide), Adam(0.1), ELBO(),
                    RBFKernel(), num_particles=100)
state, loss = einstein.train(rng_key, 15000, X_train, y_train,
                             callbacks=[Progbar()])
```

Prediction

```
y_pred = einstein.predict(state, X_test)['y']
print(f"Accuracy: {np.mean(y_pred == y_test)*100:.2f} %")
```

Einstein VI is a NumPyro (Phan 2019) library that integrates the latest developments of Stein VI (Liu and Wang 2016):

- ✓ Compositional library supporting various loss functions (ELBO, Rényi ELBO, custom loss, etc.), automatic marginalization of discrete variables (Obermeyer et al 2019), and deep learning.
- ✓ Inference based on ELBO-within-Stein core (Nalisnick 2017) with support for non-linear optimization (Wang and Liu 2019), a wealth of kernels that include matrix-valued ones (Wang et al. 2019) for graphical models and higher-order optimization.
- ✓ Learnable transforms on the parameter space, including triangular transforms (Parno and Marzouk 2018) and neural transforms (Hoffman et al. 2018)

Integrated Stein VI Theory

Bayesian Variational Inference

- **Goal:** Given prior $p(z)$ over model parameters z and likelihood $p(x|z)$ over data x infer posterior:
  $$p(z|x) = Z^{-1}p(x|z)p(z)$$
- **Problem:** Normalization constant $Z = \int p(x|z)\,dx$ is intractable.
- **Solution**: Variational Inference
  - Posit easy to evaluate approximate distribution $q(z)$
  - Minimize the Kullback-Leibler (KL) divergence $D_{\mathrm{KL}}(q \parallel p)$ which can be done without explicitly calculating Z.
- **Stein VI**: VI (Liu and Wang 2016) Use particles $\{z_i\}_{i=1}^N$ for approximation, so $q(z) = \sum_i \delta_{z_i}(z)$ and minimize KL divergence by iterating Stein forces $S(z)$.

Two Forces of EinStein VI

- **Assumption**: We have a negative loss function $\mathcal{L}(\boldsymbol{\theta})$ we would like to maximize.
- **Example:** Evidence Lower Bound (ELBO; Kingma and Welling, 2014).
- **Desiderata:** We would like to have flexibility to capture multi-modality for parameters $\boldsymbol{\theta}$ and quantify uncertainty on it.
- **Idea:** Use ELBO-within-Stein inference (Nalisnick, 2017) which can be factored into an attractive force:
  $S^+(\boldsymbol{\theta}) = \mathbb{E}_{\vartheta\sim q}[k(\boldsymbol{\theta},\vartheta)\nabla_{\boldsymbol{\theta}}L(\boldsymbol{\theta})]$
  and a repulsive force:
  $S^-(\boldsymbol{\theta}) = \mathbb{E}_{\vartheta\sim q}[\nabla_{\boldsymbol{\theta}}k(\boldsymbol{\theta},\vartheta)]$
  where k: $\mathbb{R}^d\times\mathbb{R}^d \to \mathbb{R}$ is a statistical kernel like RBF.

Matrix-Valued Kernels

- **Idea:** (Wang et al. 2019) Replace scalar-valued kernel k with matrix-valued kernel K : $\mathbb{R}^d\times\mathbb{R}^d \to \mathbb{R}^{d\times d}$
- **Update:** Change the attractive force:
  $S^+(\boldsymbol{\theta}) = \mathbb{E}_{\vartheta\sim q}[K(\boldsymbol{\theta},\vartheta)\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})]$
  and repulsive force:
  $S^-(z) = \mathbb{E}_{z'\sim q(z)}[K(z,z')\nabla_z]$
- **Advantages**: Allow pre-conditioning using second-order matrices like the Hessian or Fisher information. Allow factorization of K into local kernels $\{K^{(\ell)}\}_\ell$ based on independencies given by a graphical model.

Non-linear Stein

- **Idea:** (Wang and Liu 2019) Allow scaling of repulsive force $S^-(\boldsymbol{\theta})$ by constant

Implementation Challenges

Algorithm

Input: Classical parameters $\boldsymbol{\phi}$ and $\boldsymbol{\phi}'$, Stein parameters $\{\boldsymbol{\theta}_i\}_i$, model $p_{\boldsymbol{\phi}}(\mathbf{z},\mathbf{x})$. guide $q_{\boldsymbol{\theta},\boldsymbol{\phi}'}(\mathbf{z})$, loss $\mathcal{L}$, kernel interface KI.
Output: Parameter changes based on classical VI ($\Delta\boldsymbol{\phi}$, $\Delta\boldsymbol{\phi}'$) and Stein VI forces ($\{\Delta\boldsymbol{\theta}_i\}_i$).
$\Delta\boldsymbol{\phi} \leftarrow \mathbb{E}_{\boldsymbol{\theta}}[\nabla_{\boldsymbol{\phi}}\mathcal{L}(p_{\boldsymbol{\phi}}, q_{\boldsymbol{\theta},\boldsymbol{\phi}'})]$
$\Delta\boldsymbol{\phi}' \leftarrow \mathbb{E}_{\boldsymbol{\theta}}[\nabla_{\boldsymbol{\phi}'}\mathcal{L}(p_{\boldsymbol{\phi}}, q_{\boldsymbol{\theta},\boldsymbol{\phi}'})]$
$\{\mathbf{a}_i\}_i \leftarrow \textsc{PyTreeFlatten}(\{\boldsymbol{\theta}_i\}_i)$
$k \leftarrow \mathrm{KI}(\{\mathbf{a}_i\}_i)$
$\{\Delta\mathbf{a}_i\}_i \leftarrow \textsc{VMap}(\{\mathbf{a}_i\}_i, \mathbf{a}_i \mapsto \sum_{\mathbf{a}_j} k(\mathbf{a}_j,\mathbf{a}_i)\nabla_{\mathbf{a}_i}\mathcal{L}(p_{\boldsymbol{\phi}}, q_{\textsc{PyTreeRestore}(\mathbf{a}),\boldsymbol{\phi}'}) + \nabla_{\mathbf{a}_i}k(\mathbf{a}_j,\mathbf{a}_i))$
$\{\Delta\boldsymbol{\theta}_i\}_i \leftarrow \textsc{PyTreeRestore}(\{\Delta\mathbf{a}_i\}_i)$
return $\Delta\phi$, $\Delta\phi'$, $\{\Delta\boldsymbol{\theta}_i\}_i$

Parameter Transforms

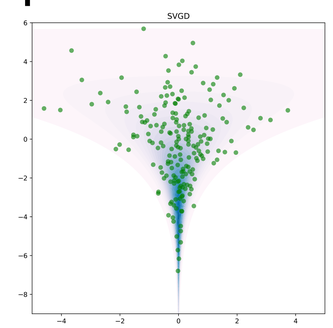
Assumption: We can write parameters for loss $\mathcal{L}_\theta$ as $\theta = \mathcal{T}(\phi)$.
Idea: We can reparametrize the Stein force:
$$S_\Phi(\phi) = S_\Theta\big(\mathcal{T}(\phi)\big)(\nabla_\phi\mathcal{T}(\phi))^\top$$
NumPyro code example on the right

```
def guide(n_flows=3, h_dim=[2,2]):
  flows = flows(n_flows, 2, h_dim)
  particle = param('p', array([0.,0.]),
                   ComposeTrans(flows))
  numpyro.sample('x', Delta(particle))
```

Implementing EinStein VI required dealing with the following challenges:

- **Re-initialiable guides** We need to re-run the initialization procedure for each Stein particle, so that they get different values for inference. `WrappedGuide` in the intro code example allows this support.
- **NumPyro and JAX integration** We rely on NumPyro for writing the probabilistic programs and integrated with their interface. The algorithm (left) needs JAX functionality such as `vmap` to calculate particle loss in parallel without changing the user code and PyTrees to map back and forth between parameters encoded as Python structures and their monolithic vectorized particle form that can capture correlation between particles.
- **Kernels** We needed a modular interface to support new kernels, based on particle information and log-probability.

Experiments and Results



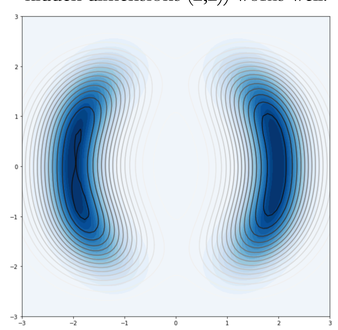
Neal's Funnel

$$y\sim\mathcal{N}(0,3) \quad x\sim\mathcal{N}(0,\exp(\tfrac{y}{2}))$$

- **Problem:** As y becomes negative, x becomes harder to sample using MCMC.
- **Result:** EinStein VI can be seen to perform well in the above figure!

Double Moon Model

- **Problem:** Multi-modal distribution which can be hard to fit using traditional VI
- **Result**: EinStein VI with neural transform (3 autoregressive flows of hidden dimensions (2,2)) works well!



Stein Mixture Latent Dirichlet Allocation (SM-LDA)

- **Goal:** Infer topics $\boldsymbol{\theta}$ from document represented as bags of words $\boldsymbol{w}$. Each document is generated according to the following process:
  $\boldsymbol{\theta} \sim \mathrm{Dir}(\alpha) \quad z_n\sim^{\mathrm{iid}} \mathrm{Cat}(\boldsymbol{\theta}) \quad w_n\sim^{\mathrm{iid}} \mathrm{Cat}(\varphi_{z_n}) \quad n\in\{1..N\}$
- **Problem:** Discrete latent variable z_n makes model non-differentiable
- **Implementation:** MLP (100 hidden dimensions), 20 topics, 5 Stein particles
- **Result:** EinStein VI works well with automatic marginalization provided by NumPyro (Obermeyer et al. 2019)

Stein Mixture Deep Markov Model (SM-DMM)

Negative test log-likelihood for JSB Chorales dataset using SM-DMM		
Krishnan and Shalit 2017	Jankowiak and Karaletsos 2019	EinStein VI (*Ours)*
6.85	6.82	6.67

- **Goal:** Sequential model inspired by HMMs but with deep neural networks for transitions and emissions. GRU guide.
- **Problem:** High-dimensional guide, over 500.000 parameters
- **Implementation:** 5 Stein particles, Adam optimizer (lr: 1e-5)
- **Result:** EinStein VI scales well and performs better than existing results!

References

D. Phan, N. Pradhan, and M. Jankowiak. Composable effects for flexible and accelerated probabilistic programming in NumPyro. Program Transformations for Machine Learning, 2019. | Q. Liu and D. Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. NeurIPS, 2018. | E. Nalisnick. Variational inference with Stein mixtures. In Advances in Approximate Bayesian Inference, 2017. | D. Wang and Q. Liu. Nonlinear stein variational gradient descent for learning diversified mixture models. ICML 2019. | D. Wang, Z. Tang, C. Bajaj, and Q. Liu. Stein variational gradient descent with matrix-valued kernels. NeurIPS 2019, 2019. | M. D. Parno and Y. M. Marzouk. Transport map accelerated Markov Chain Monte Carlo. SIAM/ASA Journal on Uncertainty Quantification, 6(2):645–682, 2018 | M. D. Hoffman, P. Sountsov, J. Dillon, I. Langmore, D. Tran, and S. Vasudevan. Neutra-lizing bad geometry in hamiltonian monte carlo using neural transport. 2018. | D. P. Kingma and M. Welling. Auto-encoding Variational Bayes. ICLR 2014, 2014. | R. G. Krishnan, U. Shalit, and D. Sontag. Structured inference networks for nonlinear state space models. AAAI 2017, 2017. M. Jankowiak and T. Karaletsos. Pathwise derivatives for multivariate distributions. AISTATS 2019, 2019.