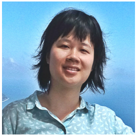
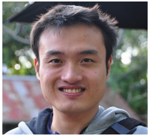
Stochastic Expectation Propagation for Large Scale Gaussian Process Classification

Daniel Hernández–Lobato[1],

Dec 11, 2015

joint work with

José Miguel Hernández-Lobato[2], Yingzhen Li[3], Thang Bui[3] and Richard E. Turner[3].

[1]Universidad Autónoma de Madrid.
[2]Harvard University.
[3]Cambridge University.

Large Scale Learning Problems

Model with **global latent variables** $\mathbf{z}$ and **hyper-parameters** $\boldsymbol{\xi}$, observed data $\mathbf{y}$ and a likelihood with N factors. We want to:

- Approximate $p(\mathbf{z}|\mathbf{y},\boldsymbol{\xi}) = \prod_{i=1}^N p(y_i|\mathbf{z},\boldsymbol{\xi})p(\mathbf{z}|\boldsymbol{\xi})/p(\mathbf{y}|\boldsymbol{\xi})$.
- Find good $\boldsymbol{\xi}$ by approximately maximizing $p(\mathbf{y}|\boldsymbol{\xi})$.

The VI approach finds **parametric** $q(\mathbf{z})$ and $\boldsymbol{\xi}$ by maximizing:

$$\log p(\mathbf{y}|\boldsymbol{\xi}) \geq \mathcal{L}(q,\boldsymbol{\xi}) = \sum_{i=1}^N \mathbb{E}_q\left[\log p(y_i|\mathbf{z},\boldsymbol{\xi})\right] - \mathrm{KL}(q||p_{\boldsymbol{\xi}}).$$

Stochastic gradients give a memory and cpu cost **independent** of N.

EP finds q by approximating each $p(y_i|\mathbf{z},\boldsymbol{\xi})$ with a **parametric** $\tilde{\phi}_i$:

$$q(\mathbf{z}) = \frac{\prod_{i=1}^N \tilde{\phi}_i(\mathbf{z})p(\mathbf{z}|\boldsymbol{\xi})}{Z_q}\,, \quad \tilde{\phi}_i = \arg\min \quad \mathrm{KL}(p(y_i|\mathbf{z},\boldsymbol{\xi})q^{\setminus i}||\tilde{\phi}_i q^{\setminus i})\,,$$

where $q^{\setminus i} \propto q/\tilde{\phi}_i$. Allows for online learning q which is very **efficient**.

Can we find $\boldsymbol{\xi}$ efficiently with EP by maximizing $Z_q \approx p(\mathbf{y}|\boldsymbol{\xi})$?

Large Scale Learning Problems

Model with **global latent variables** $\mathbf{z}$ and **hyper-parameters** $\boldsymbol{\xi}$, observed data $\mathbf{y}$ and a likelihood with N factors. We want to:

- Approximate $p(\mathbf{z}|\mathbf{y},\boldsymbol{\xi}) = \prod_{i=1}^N p(y_i|\mathbf{z},\boldsymbol{\xi})p(\mathbf{z}|\boldsymbol{\xi})/p(\mathbf{y}|\boldsymbol{\xi})$.
- Find good $\boldsymbol{\xi}$ by approximately maximizing $p(\mathbf{y}|\boldsymbol{\xi})$.

The VI approach finds **parametric** $q(\mathbf{z})$ and $\boldsymbol{\xi}$ by maximizing:

$$\log p(\mathbf{y}|\boldsymbol{\xi}) \geq \mathcal{L}(q,\boldsymbol{\xi}) = \sum_{i=1}^N \mathbb{E}_q\left[\log p(y_i|\mathbf{z},\boldsymbol{\xi})\right] - \mathrm{KL}(q||p_{\boldsymbol{\xi}}).$$

Stochastic gradients give a memory and cpu cost **independent** of N.

EP finds q by approximating each $p(y_i|\mathbf{z},\boldsymbol{\xi})$ with a **parametric** $\tilde{\phi}_i$:

$$q(\mathbf{z}) = \frac{\prod_{i=1}^N \tilde{\phi}_i(\mathbf{z})p(\mathbf{z}|\boldsymbol{\xi})}{Z_q}\,, \quad \tilde{\phi}_i = \arg\min \quad \mathrm{KL}(p(y_i|\mathbf{z},\boldsymbol{\xi})q^{\setminus i}||\tilde{\phi}_i q^{\setminus i})\,,$$

where $q^{\setminus i} \propto q/\tilde{\phi}_i$. Allows for online learning q which is very **efficient**.

Can we find ξ efficiently with EP by maximizing $Z_q \approx p(\mathbf{y}|\xi)$?

Large Scale Learning Problems

Model with **global latent variables** $\mathbf{z}$ and **hyper-parameters** $\boldsymbol{\xi}$, observed data $\mathbf{y}$ and a likelihood with N factors. We want to:

- Approximate $p(\mathbf{z}|\mathbf{y},\boldsymbol{\xi}) = \prod_{i=1}^N p(y_i|\mathbf{z},\boldsymbol{\xi})p(\mathbf{z}|\boldsymbol{\xi})/p(\mathbf{y}|\boldsymbol{\xi})$.
- Find good $\boldsymbol{\xi}$ by approximately maximizing $p(\mathbf{y}|\boldsymbol{\xi})$.

The VI approach finds **parametric** $q(\mathbf{z})$ and $\boldsymbol{\xi}$ by maximizing:

$$\log p(\mathbf{y}|\boldsymbol{\xi}) \geq \mathcal{L}(q,\boldsymbol{\xi}) = \sum_{i=1}^N \mathbb{E}_q\left[\log p(y_i|\mathbf{z},\boldsymbol{\xi})\right] - \mathrm{KL}(q||p_{\boldsymbol{\xi}}).$$

Stochastic gradients give a memory and cpu cost **independent** of N.

EP finds q by approximating each $p(y_i|\mathbf{z},\boldsymbol{\xi})$ with a **parametric** $\tilde{\phi}_i$:

$$q(\mathbf{z}) = \frac{\prod_{i=1}^N \tilde{\phi}_i(\mathbf{z})p(\mathbf{z}|\boldsymbol{\xi})}{Z_q}\,, \quad \tilde{\phi}_i = \arg\min \quad \mathrm{KL}(p(y_i|\mathbf{z},\boldsymbol{\xi})q^{\setminus i}||\tilde{\phi}_i q^{\setminus i})\,,$$

where $q^{\setminus i} \propto q/\tilde{\phi}_i$. Allows for online learning q which is very **efficient**.

Can we find ξ efficiently with EP by maximizing $Z_q \approx p(\mathbf{y}|\xi)$?

Large Scale Learning Problems

Model with **global latent variables** $\mathbf{z}$ and **hyper-parameters** $\boldsymbol{\xi}$, observed data $\mathbf{y}$ and a likelihood with N factors. We want to:

- Approximate $p(\mathbf{z}|\mathbf{y},\boldsymbol{\xi}) = \prod_{i=1}^N p(y_i|\mathbf{z},\boldsymbol{\xi})p(\mathbf{z}|\boldsymbol{\xi})/p(\mathbf{y}|\boldsymbol{\xi})$.
- Find good $\boldsymbol{\xi}$ by approximately maximizing $p(\mathbf{y}|\boldsymbol{\xi})$.

The VI approach finds **parametric** $q(\mathbf{z})$ and $\boldsymbol{\xi}$ by maximizing:

$$\log p(\mathbf{y}|\boldsymbol{\xi}) \geq \mathcal{L}(q,\boldsymbol{\xi}) = \sum_{i=1}^N \mathbb{E}_q\left[\log p(y_i|\mathbf{z},\boldsymbol{\xi})\right] - \mathrm{KL}(q||p_{\boldsymbol{\xi}}).$$

Stochastic gradients give a memory and cpu cost **independent** of N.

EP finds q by approximating each $p(y_i|\mathbf{z},\boldsymbol{\xi})$ with a **parametric** $\tilde{\phi}_i$:

$$q(\mathbf{z}) = \frac{\prod_{i=1}^N \tilde{\phi}_i(\mathbf{z})p(\mathbf{z}|\boldsymbol{\xi})}{Z_q}\,, \quad \tilde{\phi}_i = \arg\min \quad \mathrm{KL}(p(y_i|\mathbf{z},\boldsymbol{\xi})q^{\setminus i}||\tilde{\phi}_i q^{\setminus i})\,,$$

where $q^{\setminus i} \propto q/\tilde{\phi}_i$. Allows for online learning q which is very **efficient**.

Can we find $\boldsymbol{\xi}$ efficiently with EP by maximizing $Z_q \approx p(\mathbf{y}|\boldsymbol{\xi})$?

Hyper-parameter Learning in Expectation Propagation

At **convergence**, the gradient of Z_q w.r.t. each ξ_j is (Seeger, 2006):

$$\frac{\partial \log Z_q}{\partial \xi_j} = \underbrace{(\eta - \eta_{\text{prior}})^{\text{T}} \frac{\partial \theta_{\text{prior}}}{\partial \xi_j}}_{\text{Mismatch between } q \text{ and } p_{\boldsymbol{\xi}}} + \underbrace{\sum_{i=1}^{N} \frac{\partial \log \mathbb{E}_{q^{\setminus i}}[p(y_i|\mathbf{z}, \boldsymbol{\xi})]}{\partial \xi_j}}_{\text{Likelihood contribution}}$$

where η, η_{prior} are moments and θ, θ_{prior} are natural parameters.

Can we do **more frequent updates** of the hyper-parameters?

Yes! Take a gradient step on Z_q after each complete update of **all** $\tilde{\phi}_i$.

(Hernández-Lobato & Hernández-Lobato, 2015)

Hyper-parameter Learning in Expectation Propagation

At **convergence**, the gradient of Z_q w.r.t. each ξ_j is (Seeger, 2006):

$$\frac{\partial \log Z_q}{\partial \xi_j} = \underbrace{(\eta - \eta_{\text{prior}})^{\text{T}} \frac{\partial \theta_{\text{prior}}}{\partial \xi_j}}_{\text{Mismatch between } q \text{ and } p_{\boldsymbol{\xi}}} + \underbrace{\sum_{i=1}^{N} \frac{\partial \log \mathbb{E}_{q^{\setminus i}}[p(y_i|\mathbf{z}, \boldsymbol{\xi})]}{\partial \xi_j}}_{\text{Likelihood contribution}}$$

where η, η_{prior} are moments and θ, θ_{prior} are natural parameters.

Can we do **more frequent updates** of the hyper-parameters?

Yes! Take a gradient step on Z_q after each complete update of **all** $\tilde{\phi}_i$.

(Hernández-Lobato & Hernández-Lobato, 2015)

Hyper-parameter Learning in Expectation Propagation

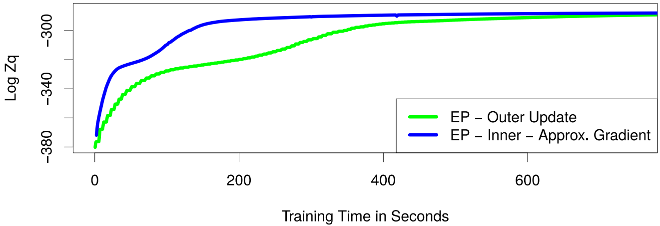
At **convergence**, the gradient of Z_q w.r.t. each ξ_j is (Seeger, 2006):

$$\frac{\partial \log Z_q}{\partial \xi_j} = \underbrace{(\eta - \eta_{\text{prior}})^{\text{T}} \frac{\partial \theta_{\text{prior}}}{\partial \xi_j}}_{\text{Mismatch between } q \text{ and } p_{\boldsymbol{\xi}}} + \underbrace{\sum_{i=1}^{N} \frac{\partial \log \mathbb{E}_{q^{\setminus i}}[p(y_i|\mathbf{z}, \boldsymbol{\xi})]}{\partial \xi_j}}_{\text{Likelihood contribution}}$$

where η, η_{prior} are moments and θ, θ_{prior} are natural parameters.

Can we do **more frequent updates** of the hyper-parameters?

Yes! Take a gradient step on Z_q after each complete update of **all** $\tilde{\phi}_i$.



(Hernández-Lobato & Hernández-Lobato, 2015)

EP Algorithm with Stochastic Gradients

Stochastic estimate of the gradient using a mini-batch $\mathcal{M}_k$:

$$\frac{\partial \log Z_q}{\partial \xi_j} \approx (\eta - \eta_{\text{prior}})^{\text{T}} \frac{\partial \theta_{\text{prior}}}{\partial \xi_j} + \frac{N}{|\mathcal{M}_k|} \sum_{i \in \mathcal{M}_k} \frac{\partial \log \mathbb{E}_{q^{\setminus i}}[p(y_i|\mathbf{z}, \boldsymbol{\xi})]}{\partial \xi_j}$$

Allows for more frequent hyper-parameter updates!

EP algorithm with mini-batches:

1. $\forall i \in \mathcal{M}_k$, update $\tilde{\phi}_i$.
2. Reconstruct the approximation q.
3. Compute a noisy estimate of the gradient of $\log Z_q$ w.r.t. each ξ_j.
4. Update all hyper-parameters ξ_j.
5. Reconstruct the approximation q.

The training cost is independent of the training set size N.

The memory resources scale with the training set size N.

EP Algorithm with Stochastic Gradients

Stochastic estimate of the gradient using a mini-batch $\mathcal{M}_k$:

$$\frac{\partial \log Z_q}{\partial \xi_j} \approx (\eta - \eta_{\text{prior}})^{\text{T}} \frac{\partial \theta_{\text{prior}}}{\partial \xi_j} + \frac{N}{|\mathcal{M}_k|} \sum_{i \in \mathcal{M}_k} \frac{\partial \log \mathbb{E}_{q^{\setminus i}}[p(y_i|\mathbf{z}, \boldsymbol{\xi})]}{\partial \xi_j}$$

Allows for more frequent hyper-parameter updates!

EP algorithm with mini-batches:

1. $\forall i \in \mathcal{M}_k$, update $\tilde{\phi}_i$.
2. Reconstruct the approximation q.
3. Compute a noisy estimate of the gradient of $\log Z_q$ w.r.t. each ξ_j.
4. Update all hyper-parameters ξ_j.
5. Reconstruct the approximation q.

The training cost is independent of the training set size N.

The memory resources scale with the training set size N.

EP Algorithm with Stochastic Gradients

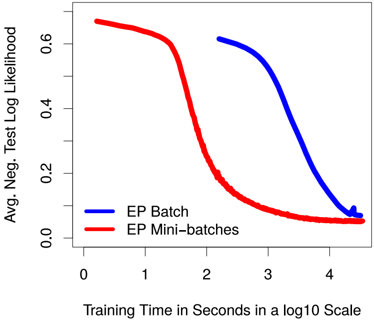
Stochastic estimate of the gradient using a mini-batch $\mathcal{M}_k$:

$$\frac{\partial \log Z_q}{\partial \xi_j} \approx (\eta - \eta_{\text{prior}})^{\text{T}} \frac{\partial \theta_{\text{prior}}}{\partial \xi_j} + \frac{N}{|\mathcal{M}_k|} \sum_{i \in \mathcal{M}_k} \frac{\partial \log \mathbb{E}_{q^{\setminus i}}[p(y_i|\mathbf{z}, \boldsymbol{\xi})]}{\partial \xi_j}$$

Allows for more frequent hyper-parameter updates!

EP algorithm with mini-batches:

1. $\forall i \in \mathcal{M}_k$, update $\tilde{\phi}_i$.
2. Reconstruct the approximation q.
3. Compute a noisy estimate of the gradient of $\log Z_q$ w.r.t. each ξ_j.
4. Update all hyper-parameters ξ_j.
5. Reconstruct the approximation q.



The training cost is independent of the training set size N.

The memory resources scale with the training set size N.

EP Algorithm with Stochastic Gradients

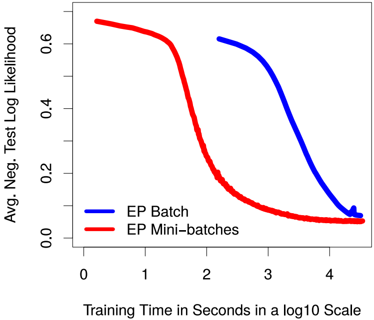
Stochastic estimate of the gradient using a mini-batch $\mathcal{M}_k$:

$$\frac{\partial \log Z_q}{\partial \xi_j} \approx (\eta - \eta_{\text{prior}})^{\text{T}} \frac{\partial \theta_{\text{prior}}}{\partial \xi_j} + \frac{N}{|\mathcal{M}_k|} \sum_{i \in \mathcal{M}_k} \frac{\partial \log \mathbb{E}_{q^{\setminus i}}[p(y_i|\mathbf{z}, \boldsymbol{\xi})]}{\partial \xi_j}$$

Allows for more frequent hyper-parameter updates!

EP algorithm with mini-batches:

1. $\forall i \in \mathcal{M}_k$, update $\tilde{\phi}_i$.
2. Reconstruct the approximation q.
3. Compute a noisy estimate of the gradient of $\log Z_q$ w.r.t. each ξ_j.
4. Update all hyper-parameters ξ_j.
5. Reconstruct the approximation q.



The training cost is independent of the training set size N.

The memory resources scale with the training set size N.

EP Algorithm with Stochastic Gradients

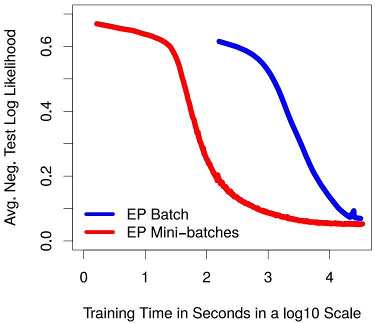
Stochastic estimate of the gradient using a mini-batch $\mathcal{M}_k$:

$$\frac{\partial \log Z_q}{\partial \xi_j} \approx (\eta - \eta_{\text{prior}})^{\text{T}} \frac{\partial \theta_{\text{prior}}}{\partial \xi_j} + \frac{N}{|\mathcal{M}_k|} \sum_{i \in \mathcal{M}_k} \frac{\partial \log \mathbb{E}_{q^{\setminus i}}[p(y_i|\mathbf{z}, \boldsymbol{\xi})]}{\partial \xi_j}$$

Allows for more frequent hyper-parameter updates!

EP algorithm with mini-batches:

1. $\forall i \in \mathcal{M}_k$, update $\tilde{\phi}_i$.
2. Reconstruct the approximation q.
3. Compute a noisy estimate of the gradient of $\log Z_q$ w.r.t. each ξ_j.
4. Update all hyper-parameters ξ_j.
5. Reconstruct the approximation q.



The training cost is independent of the training set size N.

The memory resources scale with the training set size N.

Stochastic Expectation Propagation (Li et al., 2015)

Stores only the **product of all approx. factors** $\tilde{\phi} = \prod_{i=1}^{N} \tilde{\phi}_i$.

Memory cost **independent** of the training set size N.

The EP update minimizes $\mathrm{KL}(\phi_i q^{\setminus i} || \tilde{\phi}_i q^{\setminus i})$.
Cavity distribution $q^{\setminus i}$ computation:

- **EP**: $q^{\setminus i} \propto q / \tilde{\phi}_i$.
- **SEP**: $q^{\setminus i} \propto q / \tilde{\phi}^{\frac{1}{N}}$.
- **ADF**: $q^{\setminus i} = q$.

ADF underestimates the variance!

Same updates for the **hyper-parameters** using the new cavity $q^{\setminus i}$.

Stochastic Expectation Propagation (Li et al., 2015)

Stores only the **product of all approx. factors** $\tilde{\phi} = \prod_{i=1}^{N} \tilde{\phi}_i$.

Memory cost **independent** of the training set size N.

The EP update minimizes $\mathrm{KL}(\phi_i q^{\backslash i} || \tilde{\phi}_i q^{\backslash i})$.
Cavity distribution $q^{\backslash i}$ computation:

- **EP**: $q^{\backslash i} \propto q / \tilde{\phi}_i$.
- **SEP**: $q^{\backslash i} \propto q / \tilde{\phi}^{\frac{1}{N}}$.
- **ADF**: $q^{\backslash i} = q$.

ADF underestimates the variance!

Same updates for the **hyper-parameters** using the new cavity $q^{\backslash i}$.

Stochastic Expectation Propagation (Li et al., 2015)

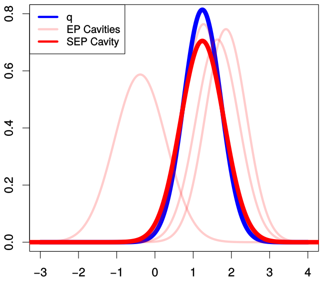
Stores only the **product of all approx. factors** $\tilde{\phi} = \prod_{i=1}^{N} \tilde{\phi}_i$.

Memory cost **independent** of the training set size N.

The EP update minimizes $\mathrm{KL}(\phi_i q^{\setminus i} || \tilde{\phi}_i q^{\setminus i})$.
Cavity distribution $q^{\setminus i}$ computation:

- **EP**: $q^{\setminus i} \propto q / \tilde{\phi}_i$.
- **SEP**: $q^{\setminus i} \propto q / \tilde{\phi}^{\frac{1}{N}}$.
- **ADF**: $q^{\setminus i} = q$.

ADF underestimates the variance!



Same updates for the **hyper-parameters** using the new cavity $q^{\setminus i}$.

Stochastic Expectation Propagation (Li et al., 2015)

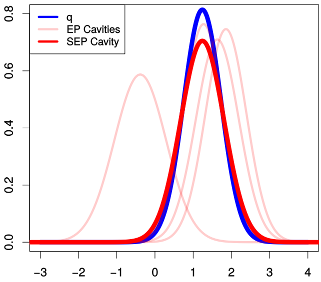
Stores only the **product of all approx. factors** $\tilde{\phi} = \prod_{i=1}^{N} \tilde{\phi}_i$.

Memory cost **independent** of the training set size N.

The EP update minimizes $\mathrm{KL}(\phi_i q^{\setminus i} || \tilde{\phi}_i q^{\setminus i})$.
Cavity distribution $q^{\setminus i}$ computation:

- **EP**: $q^{\setminus i} \propto q / \tilde{\phi}_i$.
- **SEP**: $q^{\setminus i} \propto q / \tilde{\phi}^{\frac{1}{N}}$.
- **ADF**: $q^{\setminus i} = q$.

ADF underestimates the variance!



Same updates for the **hyper-parameters** using the new cavity $q^{\setminus i}$.

Results: Sparse Gaussian Process Classification

- The latent variables $\mathbf{z}$ are the values $\overline{\mathbf{f}}$ at M **inducing points** $\overline{\mathbf{X}}$.
- $\boldsymbol{\xi}$ include $\overline{\mathbf{X}}$ and the params of the **covariance function** $k(\cdot,\cdot)$.

UCI Datasets: Batch Training.

Avg. neg. test log. likelihood			
	$M = 15\%$		
Problem	**ADF**	**EP**	**SEP**
Australian	.70± .07	.69 ± .07	**.63** ± **.05**
Breast	.12± .06	**.11** ± **.05**	**.11** ± **.05**
Crabs	.08± .06	**.06** ± **.06**	**.06** ± **.07**
Heart	.45± .18	.40 ± .13	**.39** ± **.11**
Ionosphere	.29± .18	**.26** ± **.19**	.28 ± .16
Pima	.52± .07	.52 ± .07	**.49** ± **.05**
Sonar	.40± .15	**.33** ± **.10**	.35 ± .11

MNIST: $N = 60,000$. Mini-batch training.

Airline: $N = 2,127,068$. Mini-batch training.

Why does ADF perform well on the MNIST and Airline datasets?

Results: Sparse Gaussian Process Classification

- The latent variables $\mathbf{z}$ are the values $\bar{\mathbf{f}}$ at M **inducing points** $\overline{\mathbf{X}}$.
- $\boldsymbol{\xi}$ include $\overline{\mathbf{X}}$ and the params of the **covariance function** $k(\cdot, \cdot)$.

UCI Datasets: Batch Training.

Avg. neg. test log. likelihood			
	$M = 15\%$		
Problem	**ADF**	**EP**	**SEP**
Australian	.70± .07	.69 ± .07	**.63 ± .05**
Breast	.12± .06	**.11 ± .05**	**.11 ± .05**
Crabs	.08± .06	**.06 ± .06**	**.06 ± .07**
Heart	.45± .18	.40 ± .13	**.39 ± .11**
Ionosphere	.29± .18	**.26 ± .19**	.28 ± .16
Pima	.52± .07	.52 ± .07	**.49 ± .05**
Sonar	.40± .15	**.33 ± .10**	.35 ± .11

MNIST: $N = 60,000$. Mini-batch training.

Airline: $N = 2,127,068$. Mini-batch training.

Why does ADF perform well on the MNIST and Airline datasets?

Results: Sparse Gaussian Process Classification

- The latent variables $\mathbf{z}$ are the values $\overline{\mathbf{f}}$ at M **inducing points** $\overline{\mathbf{X}}$.
- $\boldsymbol{\xi}$ include $\overline{\mathbf{X}}$ and the params of the **covariance function** $k(\cdot,\cdot)$.

UCI Datasets: Batch Training.

Avg. neg. test log. likelihood			
	$M = 15\%$		
Problem	**ADF**	**EP**	**SEP**
Australian	.70± .07	.69 ± .07	**.63 ± .05**
Breast	.12± .06	**.11 ± .05**	**.11 ± .05**
Crabs	.08± .06	**.06 ± .06**	**.06 ± .07**
Heart	.45± .18	.40 ± .13	**.39 ± .11**
Ionosphere	.29± .18	**.26 ± .19**	.28 ± .16
Pima	.52± .07	.52 ± .07	**.49 ± .05**
Sonar	.40± .15	**.33 ± .10**	.35 ± .11

MNIST: $N = 60,000$. Mini-batch training.

Airline: $N = 2,127,068$. Mini-batch training.

Why does ADF perform well on the MNIST and Airline datasets?

Results: Sparse Gaussian Process Classification

- The latent variables $\mathbf{z}$ are the values $\overline{\mathbf{f}}$ at M **inducing points** $\overline{\mathbf{X}}$.
- $\boldsymbol{\xi}$ include $\overline{\mathbf{X}}$ and the params of the **covariance function** $k(\cdot,\cdot)$.

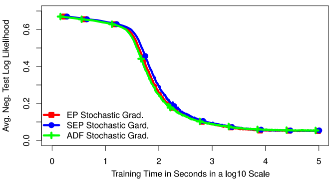
UCI Datasets: Batch Training.

Avg. neg. test log. likelihood			
	$M = 15\%$		
Problem	**ADF**	**EP**	**SEP**
Australian	.70± .07	.69 ± .07	**.63 ± .05**
Breast	.12± .06	**.11 ± .05**	**.11 ± .05**
Crabs	.08± .06	**.06 ± .06**	**.06 ± .07**
Heart	.45± .18	.40 ± .13	**.39 ± .11**
Ionosphere	.29± .18	**.26 ± .19**	.28 ± .16
Pima	.52± .07	.52 ± .07	**.49 ± .05**
Sonar	.40± .15	**.33 ± .10**	.35 ± .11

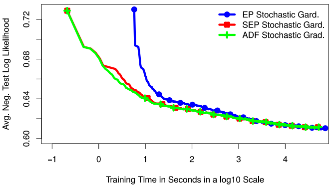
MNIST: $N = 60,000$. Mini-batch training.

Airline: $N = 2,127,068$. Mini-batch training.

MNIST odd vs even digits



Airline Delays



Why does ADF perform well on the MNIST and Airline datasets?

Results: Sparse Gaussian Process Classification

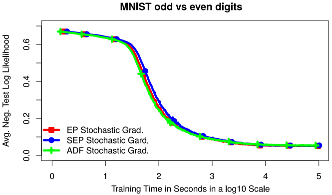
- The latent variables $\mathbf{z}$ are the values $\overline{\mathbf{f}}$ at M **inducing points** $\overline{\mathbf{X}}$.
- $\boldsymbol{\xi}$ include $\overline{\mathbf{X}}$ and the params of the **covariance function** $k(\cdot, \cdot)$.

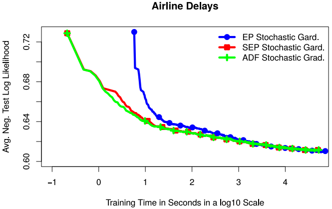
UCI Datasets: Batch Training.

Avg. neg. test log. likelihood			
	$M = 15\%$		
Problem	**ADF**	**EP**	**SEP**
Australian	.70± .07	.69 ± .07	**.63 ± .05**
Breast	.12± .06	**.11 ± .05**	**.11 ± .05**
Crabs	.08± .06	**.06 ± .06**	**.06 ± .07**
Heart	.45± .18	.40 ± .13	**.39 ± .11**
Ionosphere	.29± .18	**.26 ± .19**	.28 ± .16
Pima	.52± .07	.52 ± .07	**.49 ± .05**
Sonar	.40± .15	**.33 ± .10**	.35 ± .11

MNIST: $N = 60,000$. Mini-batch training.

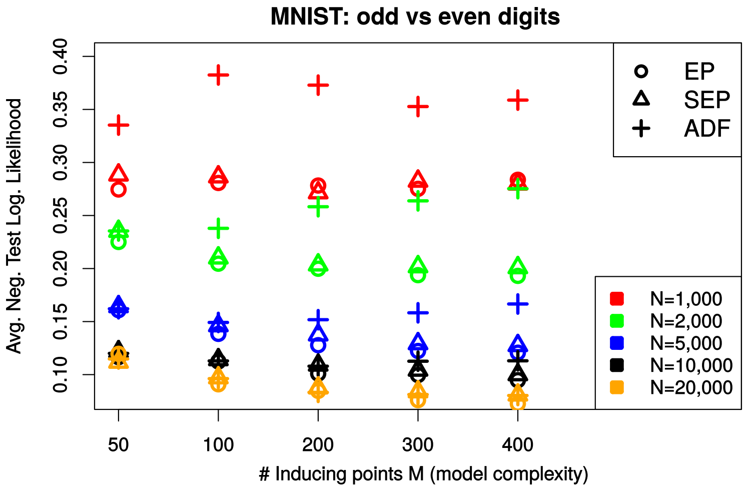
Airline: $N = 2,127,068$. Mini-batch training.





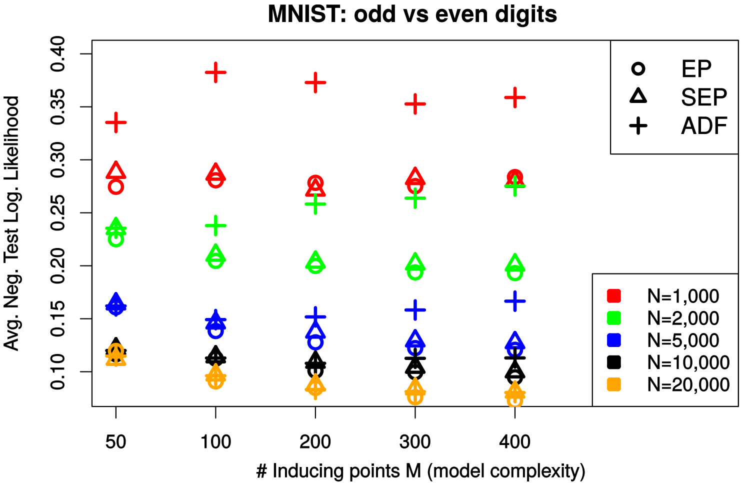
Why does ADF perform well on the MNIST and Airline datasets?

MNIST: Model Complexity vs. Number of Instances



ADF only performs well when the number of instances is very large or when the model considered is simple.

MNIST: Model Complexity vs. Number of Instances



ADF only performs well when the number of instances is very large or when the model considered is simple.

Conclusions

- It is possible to use **stochastic gradients** in expectation propagation to learn the model hyper-parameters.
- This enables using expectation propagation for approximate inference in **very large datasets**.
- The **memory cost** scales with N, since we have to store in memory the parameters of each approximate factor.
- Stochastic expectation propagation **solves** this problem without deteriorating the prediction performance!
- SEP is similar to EP in all regimes. ADF **only** when the number of instances is large and the model is small.

Conclusions

- It is possible to use **stochastic gradients** in expectation propagation to learn the model hyper-parameters.
- This enables using expectation propagation for approximate inference in **very large datasets**.
- The **memory cost** scales with N, since we have to store in memory the parameters of each approximate factor.
- Stochastic expectation propagation **solves** this problem without deteriorating the prediction performance!
- SEP is similar to EP in all regimes. ADF **only** when the number of instances is large and the model is small.

Conclusions

- It is possible to use **stochastic gradients** in expectation propagation to learn the model hyper-parameters.
- This enables using expectation propagation for approximate inference in **very large datasets**.
- The **memory cost** scales with N, since we have to store in memory the parameters of each approximate factor.
- Stochastic expectation propagation **solves** this problem without deteriorating the prediction performance!
- SEP is similar to EP in all regimes. ADF **only** when the number of instances is large and the model is small.

Conclusions

- It is possible to use **stochastic gradients** in expectation propagation to learn the model hyper-parameters.
- This enables using expectation propagation for approximate inference in **very large datasets**.
- The **memory cost** scales with N, since we have to store in memory the parameters of each approximate factor.
- Stochastic expectation propagation **solves** this problem without deteriorating the prediction performance!
- SEP is similar to EP in all regimes. ADF **only** when the number of instances is large and the model is small.

Conclusions

- It is possible to use **stochastic gradients** in expectation propagation to learn the model hyper-parameters.
- This enables using expectation propagation for approximate inference in **very large datasets**.
- The **memory cost** scales with N, since we have to store in memory the parameters of each approximate factor.
- Stochastic expectation propagation **solves** this problem without deteriorating the prediction performance!
- SEP is similar to EP in all regimes. ADF **only** when the number of instances is large and the model is small.

Thank you for your attention!

References

Hensman, James, Matthews, Alexander, and Ghahramani, Zoubin. Scalable variational gaussian process classification. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, 2015.

Hernández-Lobato, D. and Hernández-Lobato, J. M. Scalable Gaussian process classification via expectation propagation. *ArXiv e-prints*, 2015. arXiv:1507.04513.

Heskes, Tom and Zoeter, Onno. Expectation propagation for approximate inference in dynamic Bayesian networks. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence*, pp. 216–223, 2002.

Hoffman, Matthew D., Blei, David M., Wang, Chong, and Paisley, John. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

Li, Y., Hernández-Lobato, J. M., and Turner, R. Stochastic expectation propagation. In *Advances in Neural Information Processing Systems 29*, 2015.

Naish-Guzman, Andrew and Holden, Sean. The generalized fitc approximation. In *Advances in Neural Information Processing Systems 20*, pp. 1057–1064. 2008.

Qi, Yuan (Alan), Abdel-Gawad, Ahmed H., and Minka, Thomas P. Sparse-posterior gaussian processes for general likelihoods. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pp. 450–457, 2010.

Quiñonero Candela, J. and Rasmussen, C.E. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, pp. 1935–1959, 2005.

Seeger, M. Expectation propagation for exponential families. Technical report, Department of EECS, University of California, Berkeley, 2006.

Snelson, E. and Ghahramani, Z. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pp. 1257–1264, 2006.

Titsias, Michalis. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

Zeiler, Matthew D. Adadelta: An adaptive learning rate method. *ArXiv e-prints*, 2012. arXiv:1212.5701.