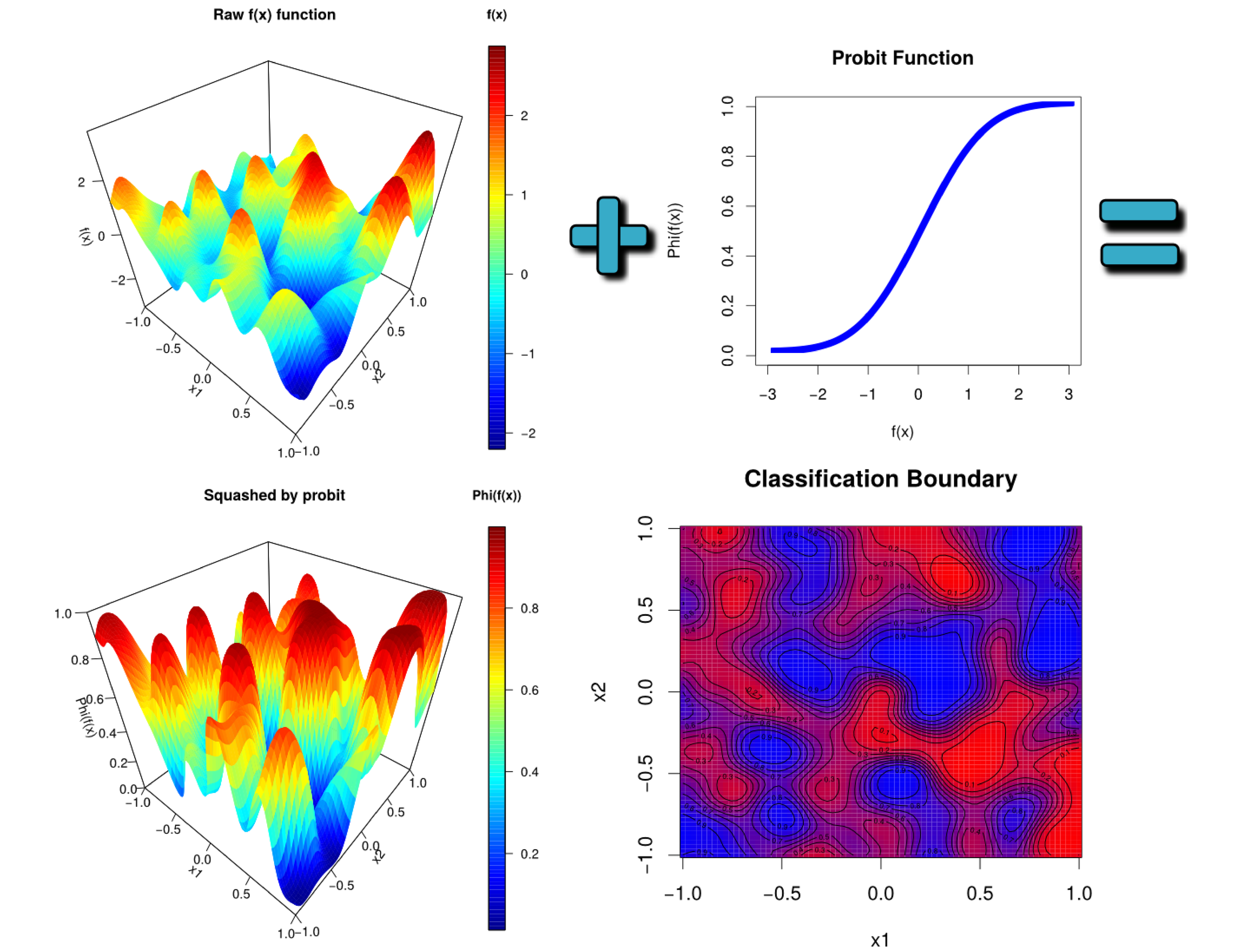
Stochastic Expectation Propagation for Large Scale Gaussian Process Classification

Daniel Hernández-Lobato[1], 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]University of Cambridge.

1. Introduction to Gaussian Process Classification (GPC)

Description: We assume $p(y_i|\mathbf{x}_i, f) = \Phi(y_i f(\mathbf{x}_i))$, where $\Phi(\cdot)$ is a probit function, $y_i \in \{-1, 1\}$, and f is generated from a **Gaussian process**, *i.e.*, $f(\mathbf{x}_i) \sim \mathcal{GP}(0, k(\mathbf{x}_i, \cdot))$, for some covariance function $k(\mathbf{x}_i, \cdot)$. Training **costs** $\mathcal{O}(n^3)$ since approximating $p(\mathbf{f}|\mathbf{y}, \mathbf{X})$ involves the inverse of a $n \times n$ matrix. Hyper-parameters are learnt via **type-II maximum likelihood**.



Non-parametric classifier that becomes more expressive as n grows!

2. Expectation Propagation (EP) for Large Scale GPC

Description: Inducing point representation in which the targets of the m inducing points are not marginalized. **Allows for very efficient training!**

$$\bar{\mathbf{X}} = (\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_m)^{\text{T}}, \qquad \bar{\mathbf{f}} = (f(\bar{\mathbf{x}}_1), \ldots, f(\bar{\mathbf{x}}_m))^{\text{T}},$$

Let $\mathbf{f} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))^{\text{T}}$. The **posterior** for $\bar{\mathbf{f}}$ is:

$$p(\bar{\mathbf{f}}|\mathbf{y}) = \frac{\int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\bar{\mathbf{f}}) p(\bar{\mathbf{f}}) d\mathbf{f}}{p(\mathbf{y})} = \frac{\prod_{i=1}^n \phi_i(\bar{\mathbf{f}}) p(\bar{\mathbf{f}})}{p(\mathbf{y})},$$

where we have used $p(\mathbf{f}|\bar{\mathbf{f}}) \approx \prod_{i=1}^n p(f_i|\bar{\mathbf{f}})$ and $\phi_i(\bar{\mathbf{f}}) = \Phi(y_i m_i / \sqrt{s_i + 1})$, with $m_i = \mathbf{K}_{f_i,\bar{\mathbf{f}}} \mathbf{K}_{\bar{\mathbf{f}},\bar{\mathbf{f}}}^{-1} \bar{\mathbf{f}}$, $s_i = \mathbf{K}_{f_i,f_i} - \mathbf{K}_{f_i,\bar{\mathbf{f}}} \mathbf{K}_{\bar{\mathbf{f}},\bar{\mathbf{f}}}^{-1} \mathbf{K}_{\bar{\mathbf{f}},f_i}$.

The posterior is approximated using **Expectation Propagation**:

$$p(\bar{\mathbf{f}}|\mathbf{y}) \approx q(\bar{\mathbf{f}}) = \frac{\prod_{i=1}^n \tilde{\phi}_i(\bar{\mathbf{f}}) p(\bar{\mathbf{f}})}{Z_q}, \qquad \tilde{\phi}_i = \arg\min \ \text{KL}(\phi_i q^{\setminus i} | \tilde{\phi}_i q^{\setminus i}),$$

where $\tilde{\phi}_i(\bar{\mathbf{f}}) = \tilde{s}_i \exp\left\{-0.5 \tilde{\nu}_i \bar{\mathbf{f}}^{\text{T}} \upsilon_i \upsilon_i^{\text{T}} \bar{\mathbf{f}} + \tilde{\mu}_i \bar{\mathbf{f}}^{\text{T}} \upsilon_i\right\}$ and $\upsilon_i = \mathbf{K}_{f_i,\bar{\mathbf{f}}} \mathbf{K}_{\bar{\mathbf{f}},\bar{\mathbf{f}}}^{-1}$.

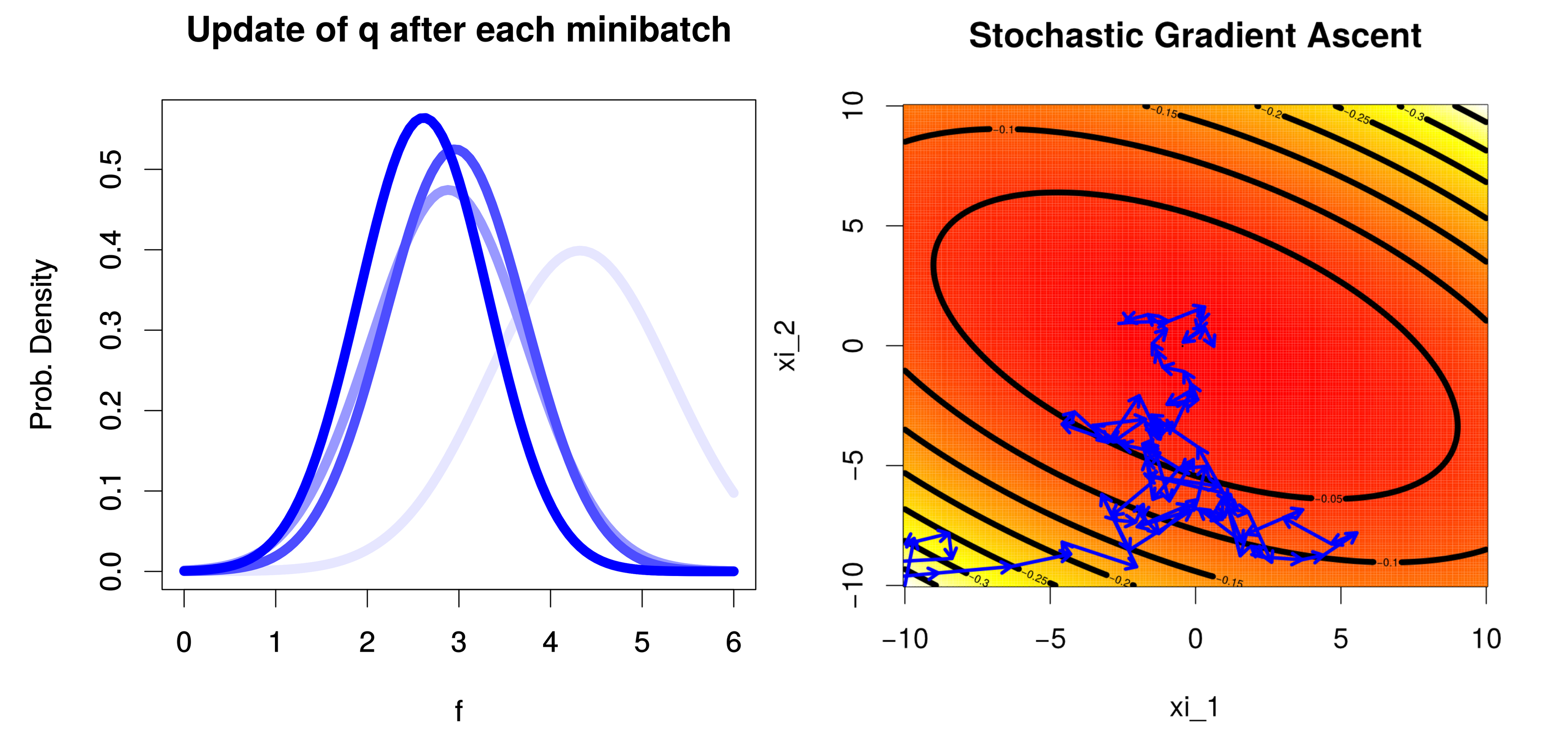
3. Learning the Hyper-parameters in Expectation Propagation

The **gradient** of log of the marginal likelihood estimate Z_q is:

$$\frac{\partial \log Z_q}{\partial \xi_j} = \eta^{\text{T}} \frac{\partial \theta_{\text{prior}}}{\partial \xi_j} - \eta_{\text{prior}}^{\text{T}} \frac{\partial \theta_{\text{prior}}}{\partial \xi_j} + \sum_{i=1}^n \frac{\partial \log Z_i}{\partial \xi_j},$$

where Z_i is the normalization constant of $\phi_i q^{\setminus i} \propto \phi_i q / \tilde{\phi}_i$, η and η_{prior} are expected sufficient statistics, and θ and θ_{prior} are natural parameters.

We can use a **minibatch** of data to refine the corresponding $\tilde{\phi}_i$ and update q, and **stochastic gradients** to maximize Z_q w.r.t the hyper-parameters ξ !



Training costs $\mathcal{O}(m^3)$ but memory resources scale like $\mathcal{O}(nm)$!

4. Stochastic Expectation Propagation (SEP)

Reduces the memory cost of EP by a factor of n. SEP uses a **single global factor** $\tilde{\phi} = \prod_{i=1}^n \tilde{\phi}_i$ to approximate the likelihood, where $\tilde{\phi}_i = (\prod_{i=1}^n \tilde{\phi}_i)^{\frac{1}{n}}$.

Algorithm: Parallel EP - Batch Mode

1: For each approximate factor $\tilde{\phi}_i$ to update:
1.1: Compute cavity: $q^{\setminus i}(\bar{\mathbf{f}}) \propto q(\bar{\mathbf{f}}) / \tilde{\phi}_i(\bar{\mathbf{f}})$
1.2: Update $\tilde{\phi}_i$: $\tilde{\phi}_i = \mathbf{proj}(\phi_i)$
2: Reconstruct q: $q(\bar{\mathbf{f}}) \propto \prod_{i=1}^n \tilde{\phi}_i(\bar{\mathbf{f}}) p(\bar{\mathbf{f}}|\bar{\mathbf{X}})$

Algorithm: Parallel ADF - Batch Mode

1: Set $q(\bar{\mathbf{f}})$ equal to the prior $p(\bar{\mathbf{f}}|\bar{\mathbf{X}})$.
2: For each exact factor ϕ_i to incorporate:
2.1: Compute cavity: $q^{\setminus i}(\bar{\mathbf{f}}) = q(\bar{\mathbf{f}})$
2.2: Find $\tilde{\phi}_i$: $\tilde{\phi}_i = \mathbf{proj}(\phi_i)$
2.3: Update q: $q(\bar{\mathbf{f}}) \propto \tilde{\phi}_i(\bar{\mathbf{f}}) q(\bar{\mathbf{f}})$

Algorithm: Parallel SEP - Batch Mode

1: Set the global factor $\tilde{\phi}_{\text{new}}$ uniform.
2: For each exact factor ϕ_i to incorporate:
2.1: Compute cavity: $q^{\setminus i}(\bar{\mathbf{f}}) \propto q(\bar{\mathbf{f}}) / \tilde{\phi}(\bar{\mathbf{f}})^{\frac{1}{n}}$
2.2: Find $\tilde{\phi}_i$: $\tilde{\phi}_i = \mathbf{proj}(\phi_i)$
2.3: Accumulate: $\tilde{\phi}_{\text{new}}(\bar{\mathbf{f}}) = \tilde{\phi}_{\text{new}}(\bar{\mathbf{f}}) \tilde{\phi}_i(\bar{\mathbf{f}})$
3: Reconstruct q: $q(\bar{\mathbf{f}}) \propto \tilde{\phi}_{\text{new}}(\bar{\mathbf{f}}) p(\bar{\mathbf{f}}|\bar{\mathbf{X}})$

In SEP all the approximate factors are the same and hence we only need to store $\mathcal{O}(m^2)$ parameters in total.

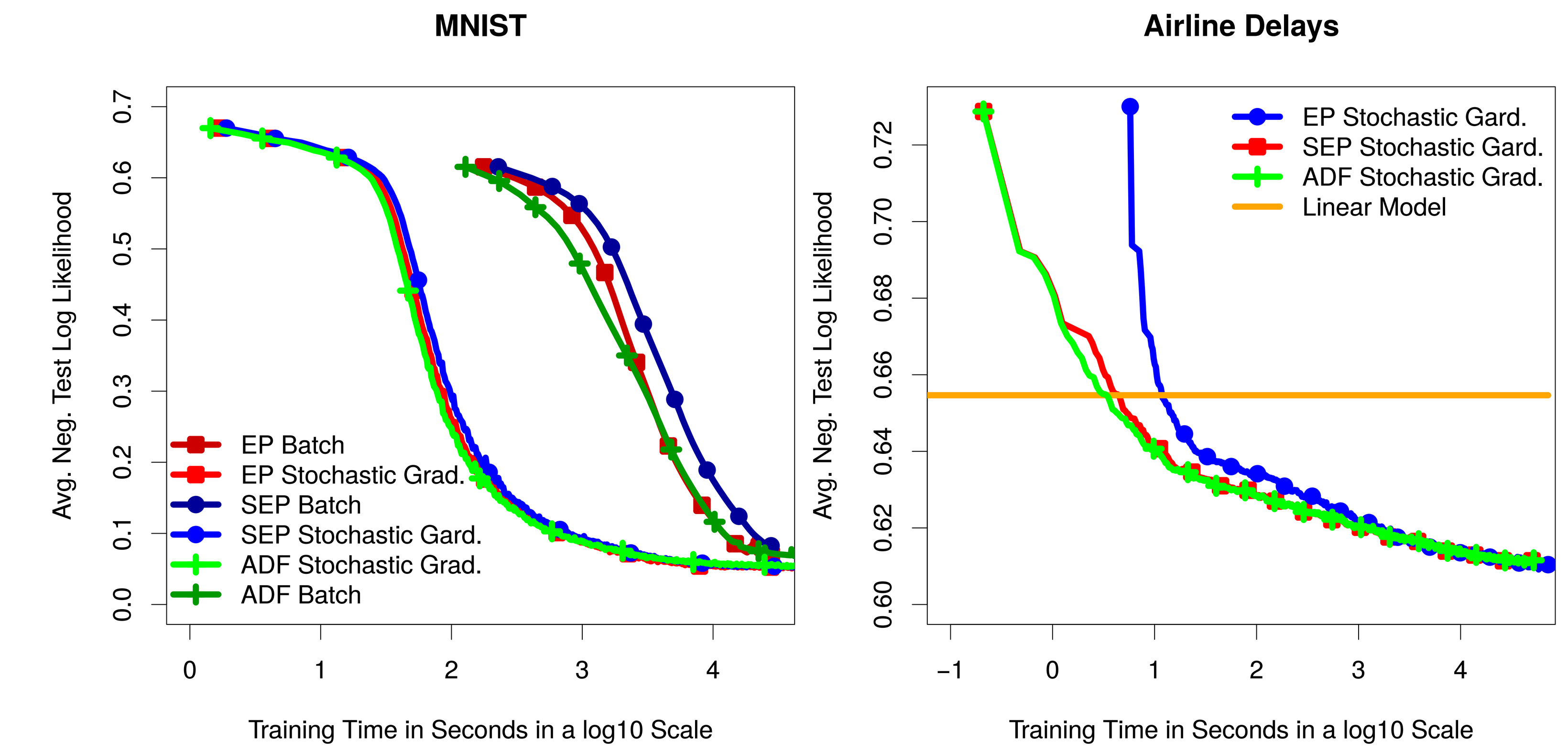
The update of the hyper-parameters is as in EP, but we have to consider the form of the cavity, *i.e.*, $q^{\setminus i} \propto q / \tilde{\phi}^{\frac{1}{n}}$, which is the same for each factor ϕ_i!

5. Experimental Results

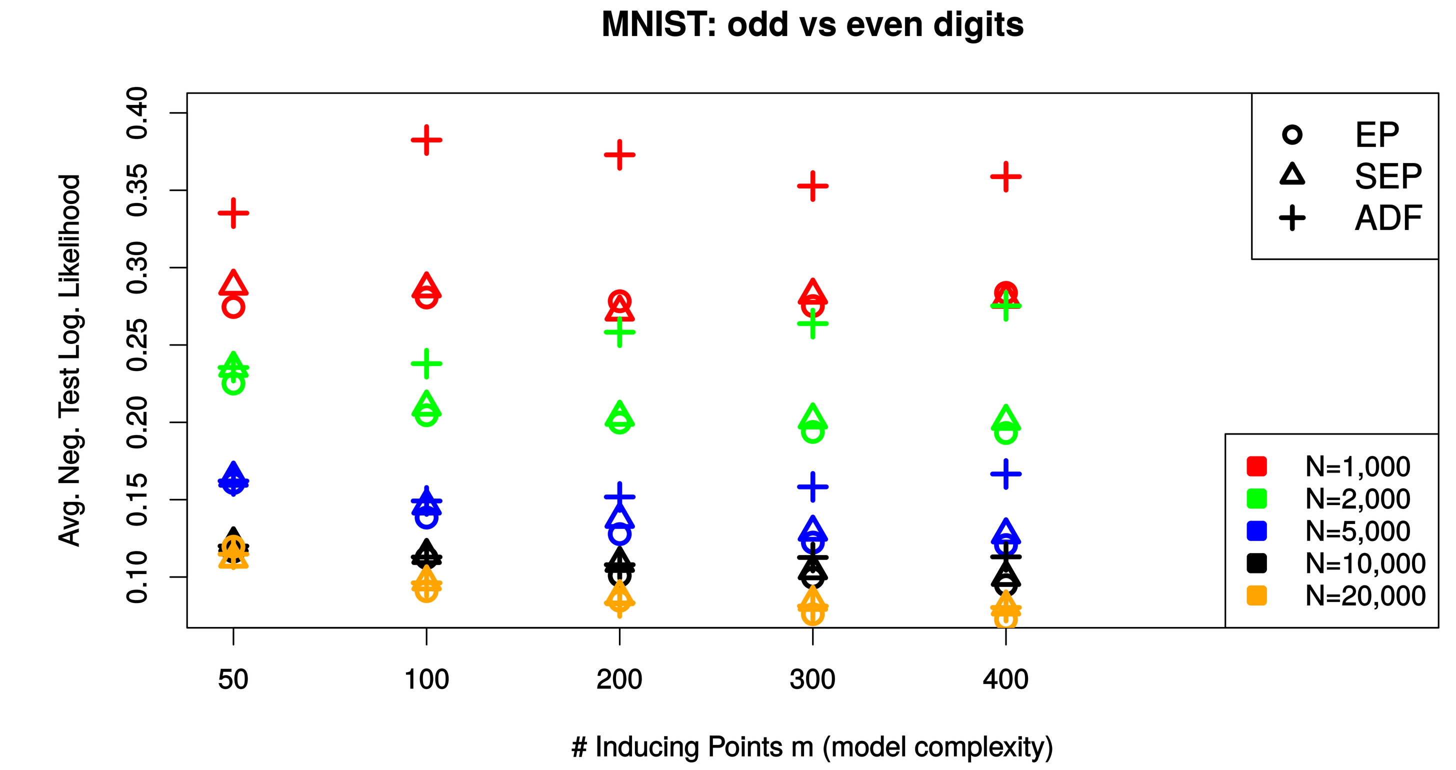
Avg. Negative test log likelihood and training time in seconds.

	$m = 15\%$			$m = 50\%$		
Problem	**ADF**	**EP**	**SEP**	**ADF**	**EP**	**SEP**
Australian	.70 ±.07	.69 ± .07	**.63 ± .05**	.67 ±.06	.64 ± .05	**.63 ± .05**
Breast	.12 ±.06	.11 ± .05	**.11 ± .05**	.12 ±.05	**.11 ± .05**	.11 ± .06
Crabs	.08 ±.06	.06 ± .06	**.06 ± .07**	.08 ±.06	.06 ± .06	**.06 ± .07**
Heart	.45 ±.18	.40 ± .13	**.39 ± .11**	.46 ±.17	.41 ± .11	**.40 ± .12**
Ionosphere	.29 ±.18	**.26 ± .19**	.28 ± .16	.33 ±.19	.27 ± .19	**.27 ± .17**
Pima	.52 ±.07	.52 ± .07	**.49 ± .05**	.62 ±.09	.50 ± .05	**.49 ± .05**
Sonar	.40 ±.15	**.33 ± .10**	.35 ± .11	.46 ±.24	**.29 ± .09**	.33 ± .12
Avg. Time	18.2±0.3	19.3± 0.5	18.8± 0.1	145±4.0	136 ± 3.0	149 ± 1.0

Number of training instances: MNIST 60,000 and Airline 2,127,068.



Why does ADF perform similar to SEP now?



6. Conclusions

- Stochastic expectation propagation (SEP) can be used as a practical alternative to expectation propagation (EP) for training Gaussian Process Classifiers on small and large datasets.
- SEP reduces the memory cost from $\mathcal{O}(nm)$ to $\mathcal{O}(m^2)$, which is very good if $n \gg m$.
- ADF also provides similar results to expectation propagation, but only when the model is simple (small m), or when the number of training instances is very large (large n).