Supplementary

S1. Graphical Models

Bayesian probability theory has played an important role in the modern machine learning and the probabilistic inference. In this section, we discuss a powerful representation of probabilistic models, called graphical models, which offer a great flexibility for problem solving and system modeling.

A graph $G(V, E)$ is defined by a set of nodes V and a set of edges E connecting these nodes, where each node represents a random variable or a group of random variables and each edge represents the statistical dependency between the connecting variables. Then the decomposition of the joint probability over all the random variables can be expressed by a graph, where the complex global algebraic calculation can be replaced by local graphical manipulations. In probabilistic graphical models, there are mainly three different kinds of graphs, i.e. directed graph, undirected graph and factor graph, where the directed and the undirected graphical models are also known as Bayesian networks and Markov random fields, respectively. In this section, we focus on the discussion of factor graph, since converting both directed and undirected graphs into factor graphs is often the canonical way for solving inference problems [15].

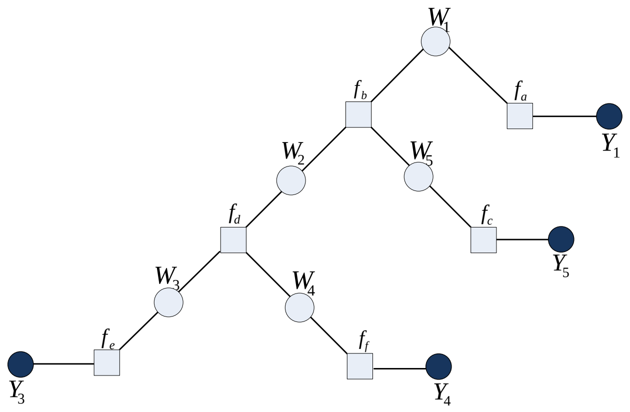

Figure S1: Example of a factor graph.

S1.1. Factor graph

Factor graph is a bipartite graph, which comprises two different kinds of nodes (i.e. a factor node and a variable node). In factor graph, each edge must connect a factor node and a variable node. Moreover, each factor node represents one of factors over subsets of some variables in a decomposed joint distribution. Each variable node expresses a random variable. Suppose that the decomposition of the joint distribution over a set of random variables has the form of a product of factors

$$p(\mathbf{w}) = \prod_s f_s(\mathbf{w}_s),$$

where $\mathbf{w}$ is a set of variables in the joint distribution, $\mathbf{w}_s$ is a subset of variables and $f_s(\mathbf{w}_s)$ is a function of all variables in $\mathbf{w}_s$.

For example, let us consider the factorization of a joint distribution $p(\mathbf{w})$ as

$$p(\mathbf{w}, \mathbf{y}) = f_a(\boldsymbol{w}_1, \mathbf{y}_1) f_b(\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_5) f_c(\boldsymbol{w}_5, \mathbf{y}_5) f_d(\boldsymbol{w}_2, \boldsymbol{w}_3, \boldsymbol{w}_4) f_e(\boldsymbol{w}_3, \mathbf{y}_3) f_c(\boldsymbol{w}_4, \mathbf{y}_4).$$

Then, Figure S1 shows the corresponding factor graph representation of the factorization of $p(\mathbf{w}, \mathbf{y})$. Please note that we use circles and squares to represent variable nodes and factor nodes in factor graph, respectively.

S2. Inference on Factor Graph

The sum-product algorithm (or Belief propagation (BP) algorithm) is an efficient and exact inference algorithm for computing local marginal over variables on tree-structured graphs. For graphs with loops, a lot of applications (e.g. Channel coding [1] and image processing [2], etc.) show that BP algorithm (or loopy BP algorithm) still provides a good performance [15].

To introduce the sum-product algorithm, let us take a look at the following example first. Suppose that a system has a set of hidden variables $\boldsymbol{w} = \{w_1, w_2, w_3, w_4, w_5\}$ and a set of observations $\mathbf{y} = \{y_1, y_3, y_4, y_5\}$. We are interested in the estimate of each hidden variable w_i, $i = 1,2,\cdots,5$, given the observed data $\mathbf{y}$. Thus, the estimate $\hat{x}_i$ can be expressed as

$$\begin{aligned}\hat{w}_i &= \operatorname*{argmax}_{w_i} p(w_i|\mathbf{y}) \\ &= \operatorname*{argmax}_{w_i} \frac{p(w_i, \mathbf{y})}{p(\mathbf{y})} \\ &= \operatorname*{argmax}_{w_i} p(w_i, \mathbf{y}).\end{aligned}$$

The above equation requires us to compute the marginal distribution $p(w_i, \mathbf{y})$ out of the joint distribution $p(\mathbf{w}, \mathbf{y})$. For this toy problem, we can compute the marginal distributions for each variable independently. However, for a large-scale problem with hundred or even thousand variables, it is infeasible to independently marginalize each variable, since the computational burden is very expensive. Fortunately, BP algorithm on factor graph provides an efficient way to compute marginal distributions over hidden variables.

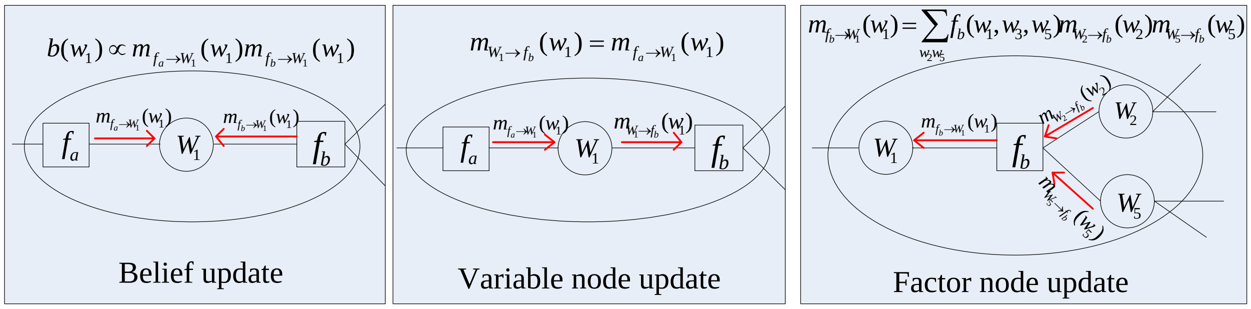

Figure S2: Example of message update in BP algorithm

Suppose that the factorization of joint probability $p(\mathbf{x}, \mathbf{y})$ takes the form

$$\begin{aligned}p(\mathbf{w}, \mathbf{y}) &= p(y_1|w_1) p(w_1, w_2, w_5) p(y_5|w_5) p(w_3, w_4|w_2) p(y_3|w_3) p(y_4|w_4) \\ &= f_a(w_1, y_1) f_b(w_1, w_2, w_5) f_c(w_5, y_5) f_d(w_2, w_3, w_4) f_e(w_3, y_3) f_c(w_4, y_4),\end{aligned}$$

where each function $f_s(\cdot)$ corresponds to a factor with some variables in the joint distribution. Please note that the above factorization can be exactly expressed by the factor graph in Figure S2. For example, let us compute the marginal probability of the discrete variable w_1 as follows

$$p(w_1, \mathbf{y}) = \underbrace{f_a(w_1, y_1)}_{m_{f_a \to w_1}(w_1)} \underbrace{\textstyle\sum_{w_1, w_5} f_b(w_1, w_2, w_5) \underbrace{f_c(w_5, y_5)}_{m_{f_c \to w_5}(w_5)} \underbrace{\textstyle\sum_{w_3, w_4} f_d(w_3, x_4, x_2) \underbrace{f_e(w_3, y_3)}_{m_{f_e \to w_3}(w_3)} \underbrace{f_f(w_4, y_4)}_{m_{f_f \to w_4}(w_4)}}_{m_{f_d \to w_2}(w_2)}}_{m_{f_b \to w_1}(w_1)}$$

where $m_{f_s \to W_i}(w_i)$ denotes a message sent from a factor node f_s to a variable node W_i. Moreover, let us introduce $m_{W_i \to f_s}(w_i)$ as the message sent from a variable node W_i to a factor node f_s (see Figure S2 for more details).
By inspecting the above equation, we can conclude that the variable node message update rule as

$$m_{W_i \to f_s}(w_i) \propto \prod_{s' \in \mathcal{N}(W_i) \backslash s} m_{f_{s'} \to W_i}(w_i),$$

and the factor node update rule as

$$m_{f_s \to W_i}(w_i) \propto \sum_{\boldsymbol{w_s} \backslash w_i} \left(f_s(\boldsymbol{w_s}) \prod_{j \in \mathcal{N}(f_s) \backslash i} m_{W_j \to f_s}(w_j) \right),$$

where $\mathcal{N}(W_i)\backslash s$ denotes the set of all neighbors' indices of node W_i excluding the index s of the factor node f_s; $f_s(\boldsymbol{w_s})$ is the factor function for the factor node f_s; $\sum_{\boldsymbol{w_s} \backslash x_i}$ denotes a sum over all the variables in $\boldsymbol{w_s}$, that are arguments of $f_s(\boldsymbol{w_s})$, except w_i. Loosely speaking, $m_{f_s \to W_i}(w_i)$ and $m_{W_i \to f_s}(w_i)$ can be interpreted as the beliefs of node W_i taking the value w_i transmitting from node f_s to W_i and vice versa. Finally, the sum-product algorithm computes the marginal probability of variable x_i, also called the belief $b(w_i)$ at a variable node W_i, as follows

$$b(w_i) \propto \prod_{s \in \mathcal{N}(W_i)} m_{f_s \to W_i}(w_i).$$

So far, we suppose that all of the variables are discrete, so the marginalization is computed by summation. However, the sum-product algorithm is also applicable to linear-Gaussian models by replacing summation by integration, e.g. Kalman filtering.

S3. Estimating Model coefficients through Maximum Likelihood (ML) Estimator

Given a training dataset $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$, where $y_i \in \{0, 1\}$ and $\boldsymbol{x_i}$ are the binary label and the feature vector of each record, respectively, with $i = 1, \ldots, m$. Moreover, let us denote by $\boldsymbol{Y} = \{y_1, \ldots y_m\}$ the set of binary labels. Then the likelihood function in the ML based LR for parameter $\boldsymbol{\beta}$ can be factorized as

$$p(\boldsymbol{Y}|\boldsymbol{\beta}) = \prod_{i=1}^{m} p(y_i|\boldsymbol{\beta}) = \prod_{i=1}^{m} \pi(\boldsymbol{\beta^T x_i})^{y_i} \left(1 - \pi(\boldsymbol{\beta^T x_i})\right)^{1-y_i},$$

since each records are conditionally independent given the coefficient β. For mathematical convenience, the maximization of the above likelihood function is usually solved in its logarithm domain through some numerical methods, e.g. Newton-Raphson method [6]. In the Newton-Raphson method, the maximum likelihood estimator (MLE) $\hat{\beta}$ for β can be recursively obtained by

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \left[\frac{\partial^2 L(\boldsymbol{\beta}^{(k)})}{\partial \boldsymbol{\beta}^{(k)} \partial \boldsymbol{\beta}^{(k)^T}} \right]^{-1} \frac{\partial L(\boldsymbol{\beta}^{(k)})}{\partial \boldsymbol{\beta}^{(k)}},$$

where $L(\boldsymbol{\beta}^{(k)}) = \sum_1^m \left[y_i \log \pi\left(\boldsymbol{\beta}^{(k)\boldsymbol{T}} \boldsymbol{x_i}\right) + (1 - y_i) \log \left(1 - \pi\left(\boldsymbol{\beta}^{(k)\boldsymbol{T}} \boldsymbol{x_i}\right)\right) \right]$ is the logarithm of the likelihood function at the k-th Newton-Raphson iteration. Although, there have been a few previous studies that work on the distributed LR and distributed privacy preserving LR [7] based on ML treatments, there are two major difficulties that still plague the practical implementation of these ordinary LR. First, the complexity of ML based RL does not scale well for the model update of a time-varying dataset with a large size, since the model update (i.e. retraining the model coefficient based on new observed data) in a MLE needs to incorporate both the new and the previously input data. Second, the Newton-Raphson method requires the synchronization among all participating sites during each update iteration (i.e., k), which dramatically reduces the flexibility of ML based distributed LR in practice.

S4. Estimating Model coefficients through Maximum a Posterior (MAP) Estimator

Bayesian inference: Bayesian theory provides a theoretical framework for reasoning with probabilities. We provide a brief introduction of the Bayesian LR. In Bayesian inference, any unknown is expressed in terms of probability, thus the estimation of a vector variable $\boldsymbol{\beta}$ corresponds to the estimation of its posterior probability $p(\boldsymbol{\beta}|\boldsymbol{Y})$. According to Bayes' rule, the posterior probability can be reformatted as,

$$p(\boldsymbol{\beta}|\boldsymbol{Y}) = \frac{p(\boldsymbol{Y},\boldsymbol{\beta})}{p(\boldsymbol{Y})} = \frac{p(\boldsymbol{\beta})p(\boldsymbol{Y}|\boldsymbol{\beta})}{p(\boldsymbol{Y})}.$$

The probability $p(\boldsymbol{Y},\boldsymbol{\beta})$ is called the joint probability, and $p(\boldsymbol{\beta})$ is called the prior probability, which captures the belief of obtaining hidden variable $\boldsymbol{\beta}$ before observing the labels $\boldsymbol{Y}$. The quantity $p(\boldsymbol{Y}|\boldsymbol{\beta})$ measures how likely the observed data $\boldsymbol{Y}$ is for different $\boldsymbol{\beta}$, and it is called the likelihood function. Moreover, $p(\boldsymbol{Y})$ can be interpreted as the normalization constant, which can be evaluated through the marginalization step as follows

$$p(\boldsymbol{Y}) = \int p(\boldsymbol{\beta})p(\boldsymbol{Y}|\boldsymbol{\beta})d\boldsymbol{\beta}.$$

The fundamental differences between Bayesian and non-Bayesian (e.g., frequentist) paradigms are the ways they deal with the unknown variable $\boldsymbol{\beta}$. Non-Bayesian paradigm always considers a fixed $\boldsymbol{\beta}$ as unknown constant, whose value is determined by, for example, maximum likelihood estimator. The statistical accuracy of estimates (i.e., confidence interval) is obtained by evaluating different datasets sampled from $p(\boldsymbol{Y})$. In contrast, Bayesian paradigm captures the known variable $\boldsymbol{\beta}$ through its posterior distribution directly. One advantage of Bayesian inference is the inclusion of prior knowledge, which can avoid some pitfalls for ML estimators. Moreover, based on new observed data, Bayesian inference offers an easier way to do online learning by taking the old posterior distribution as the new prior. Then, through Bayesian inference, one can decouple models from observed data, where the workflow of Bayesian inference has been shown in Figure S3.

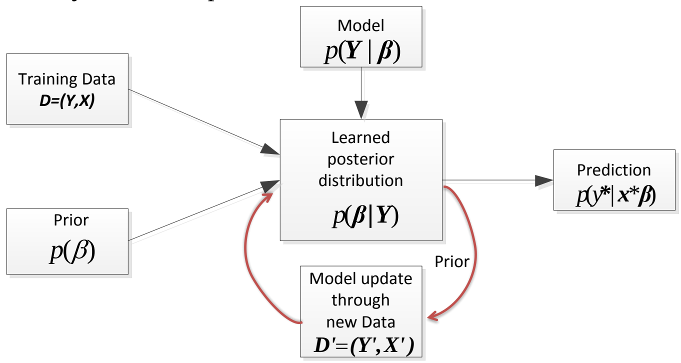

Figure S3 : Bayesian inference workflow for online learning.

Maximum a Posterior (MAP) Estimator of LR: To find the MAP solution of LR, we use the fact that all records are conditionally independent given $\boldsymbol{\beta}$ and thus factorize the posterior as

$$p(\boldsymbol{\beta}|\boldsymbol{Y}) = \frac{p(\boldsymbol{\beta})p(\boldsymbol{Y}|\boldsymbol{\beta})}{p(\boldsymbol{Y})} = \frac{1}{Z}p(\boldsymbol{\beta})\prod_{i=1}^{m} p(y_i|\boldsymbol{\beta}),$$

where $Z = p(\boldsymbol{Y}) = \int p(\boldsymbol{\beta})p(\boldsymbol{Y}|\boldsymbol{\beta})d\boldsymbol{\beta}$ is called the normalization constant. For mathematical convenience, we replace all 0's in the vector of class label $\boldsymbol{Y}$ with $\mathbf{-1}$**'s** in Bayesian LR. Therefore, within the new alphabet of $y_i \in \{-1, 1\}$, one can readily show the likelihood function as

$$p(y_i|\boldsymbol{\beta}) = \pi(y_i\boldsymbol{\beta}^T\boldsymbol{x}_i) = \frac{1}{1+e^{-y_i\boldsymbol{\beta}^T\boldsymbol{x}_i}}.$$

Without loss of generality, we model the prior probability $p(\boldsymbol{\beta})$ using a Normal distribution $N(0,\boldsymbol{V_0})$ with zero mean and variance matrix $\boldsymbol{V_0}$. By inspecting the factorized posterior probability, we can see that each likelihood function has been decoupled with others, which provides the possibility to evaluate each likelihood function in a distributed manner. However, the posterior probability $p(\boldsymbol{\beta}|\boldsymbol{Y})$ and the normalization constant Z are mathematically intractable if we want to evaluate products of logistic likelihood functions and the corresponding integral, respectively. Thus, one workaround for these difficulties is to resort to use approximation inference methods, where the original logistic likelihood function can be approximated by another tractable function.

Most existing approximation methods belong to two classes, i.e., stochastic approximation and deterministic approximation. Stochastic techniques (a.k.a., sampling methods) are more general and can be applicable to any kind of distribution. However, stochastic methods are usually computational demanding in practice. In contrast, deterministic approximation schemes provide some low complexity alternatives based on the analytical approximations to the posterior distribution. The existing deterministic approximation methods include Laplace approximation (LA), Variational Bayes (VB), Expectation Propagation (EP) and so on [8]. In this paper, we proposed an EXpectation Propagation based LOgistic REgression (EXPLORER) framework for distributed privacy preserving LR in Bayesian paradigm, since EP typically shows a higher accuracy comparing with other deterministic approximation methods, when it is designed properly.

S5. Complexity of EXPLORER vs. Ordinary LR

The most complexity parts of ML based ordinary LR are the inversion of the coefficient matrix and the multiplications among training vectors. The above two operation would result a complexity with the order of $O(d^3)$ and $O(2m^2d)$, respectively, where d is the dimension of coefficient matrix and n is number of records in a dataset. Let's denote by k the total number of iterations that an ML estimator of LR needs to converge. Then the total complexity of an ordinary LR is $O(kd^3 + 2km^2d)$. If the training dataset keeps increasing over time, an ML based LR must retrain the model by involving both the new and the previous data. Therefore, the complexity of each model update in an ordinary LR is proportional to $O(Tkd^2 + 2kd\sum_{t=1}^{T} m_t{}^2)$, where T is the total number of model updates requested by users, n_t is the dataset size in each updates. For the ease of exposition, we assume the size of the dataset increases at the rate $m_t = \sqrt{2}m_{t-1}$. Then the complexity of an ordinary LR can be expressed as $O(Tkd^3 + 2kdT^3m^2)$.

For the complexity of EXPLORER, the most complexity part is also the inversion of covariance matrix (i.e., $O(d^3)$). Since we must perform matrix inversion of each record, the total complexity is proportional to $O(kmd^3)$). If we also concern an incremental update of the dataset at the rate $\mathrm{m_t} = \sqrt{2}\mathrm{m_{t-1}}$, the complexity of each inter-site update can be express as $O(T(\sqrt{2} - 1)kmd^3)$. It is readily to show that if $d < T\sqrt{m}$, the proposed EXPLORER offers a lower complexity compared with Ordinary LR at each inter-site update step. Therefore, we conclude that the proposed EXPLORER framework is computationally favorable for online learning. Moreover, the overall complexity of EXPLORER is always α times of complexity of an inter-site update step, where α is the number of iterations used for the convergence of the inter-site update. As shown in our experiments, α is usually a very small number (e.g. 4), thus, the proposed EXPLORER only introduced a very limit overhead. It is worth mentioning that each site can parallel update. In other words, if there is n number of participant sites, the execution time of EXPLORER would be proportional to $\frac{\alpha}{n}$. Finally, it is worth mentioning that the speedup of the proposed EXPLORER is a result of distributed computing (i.e., multiple computing sites), while the overall complexity of the proposed EXPLORER depends on the data size among all sites.

References

[1] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics letters*, vol. 32, no. August, pp. 1645-1646, 1996.

[2] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient Belief Propagation for Early Vision," *International Journal of Computer Vision*, vol. 70, no. 1, pp. 41-54, May 2006.

[3] T. P. Minka, "Expectation propagation for approximate Bayesian inference," in *Uncertainty in Artificial Intelligence*, 2001, vol. 17, pp. 362-369.

[4] T. Minka, "Divergence measures and message passing," *Microsoft Research Tech Rep MSRTR2005173*, vol. 2005, pp. 1-17, 2005.

[5] H.-andrea Loeliger, “An Introduction to factor graphs,” *IEEE Signal Processing Magazine*, vol. 21, no. 1, pp. 28-41, Jan. 2004.

[6] T. Minka, “A comparison of numerical optimizers for logistic regression,” *CMU Technical Report*, vol. 2003, pp. 1-18, 2003.

[7] Y. Wu, X. Jiang, J. Kim, and L. Ohno-Machado, “Grid LOgistic REgression (GLORE): Building Shared Models Without Sharing Data,” *Journal of the American Medical Informatics Association (1st revision)*, 2012.

[8] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[9] D. W. Hosmer, T. Hosmer, S. Le Cessie, and S. Lemeshow, “A comparison of goodness-of-fit tests for the Logistic Regression model,” *Statistics in Medicine.*, vol. 16, no. 9, pp. 965-980, 1997.

[10] T. A. Lasko, J. G. Bhagwat, K. H. Zou, and L. Ohno-Machado, “The use of receiver operating characteristic curves in biomedical informatics,” *Journal of Biomedical Informatics*, vol. 38, no. 5, pp. 404-415, 2005.

[11] R. L. Kennedy, a M. Burton, H. S. Fraser, L. N. McStay, and R. F. Harrison, “Early diagnosis of acute myocardial infarction using clinical and electrocardiographic data at presentation: derivation and evaluation of logistic regression models.,” *European Heart Journal*, vol. 17, no. 8, pp. 1181-91, Aug. 1996.

[12] R. L. Kennedy, a M. Burton, H. S. Fraser, L. N. McStay, and R. F. Harrison, “Early diagnosis of acute myocardial infarction using clinical and electrocardiographic data at presentation: derivation and evaluation of logistic regression models,” *European Heart Journal*, vol. 17, no. 8, pp. 1181-1191, Aug. 1996.

[13] K. H. Zou, A. I. Liu, A. I. Bandos, L. Ohno-Machado, and H. E. Rockette, *Statistical evaluation of diagnostic performance: topics in ROC analysis*. Boca Raton, FL: CRC Press, Chapman & Hall, 2011.

[14] D. W. Hosmer, T. Hosmer, S. Le Cessie, and S. Lemeshow, “A comparison of goodness-of-fit tests for the Logistic Regression model,” *Statistics in Medicine*, vol. 16, no. 9, pp. 965-980, 1997.

[15] S. Wang, “Adaptive Channel and Source Coding Using Approximate Inference”, Ph.D. Dissertation, 2011

[16] Thomas Minka, “EP: A quick reference”, Technical Report 2008