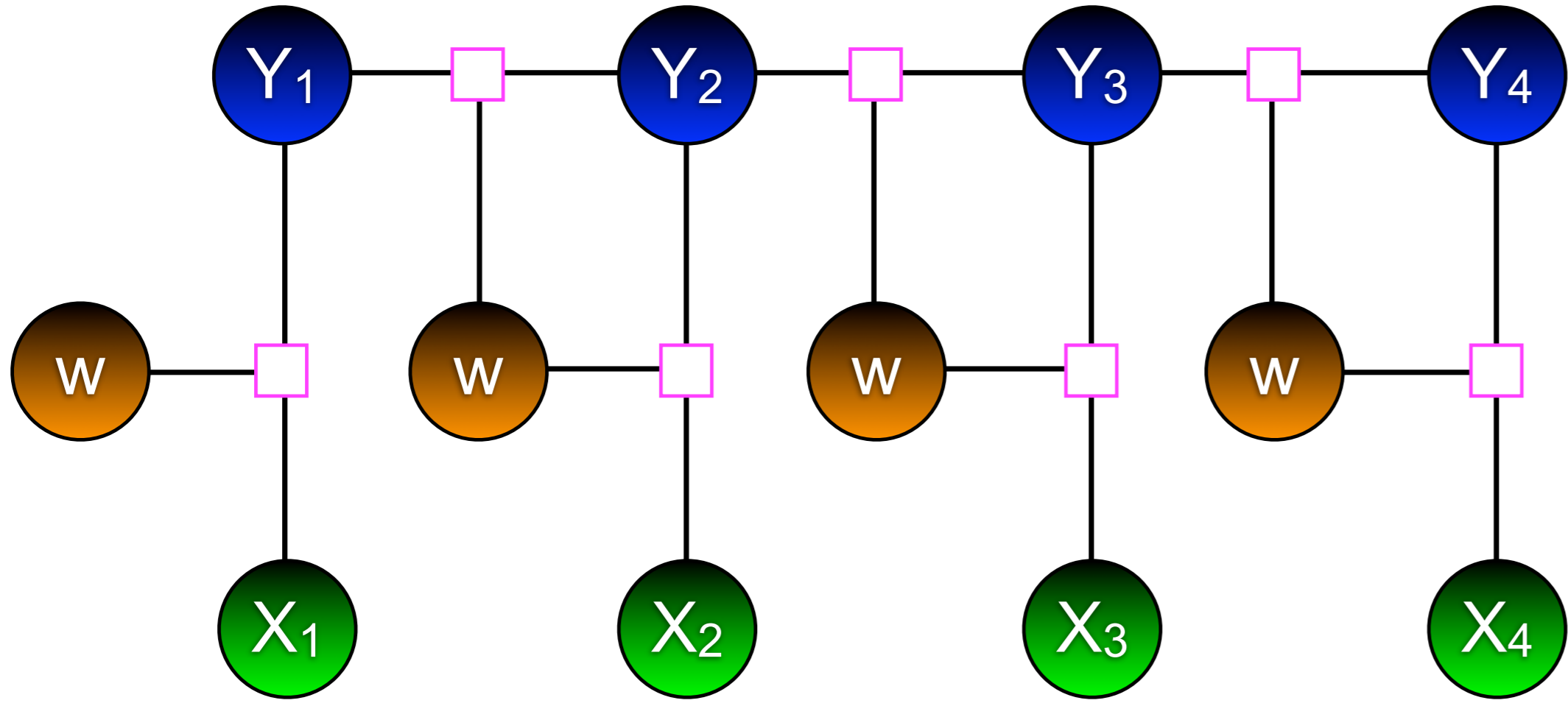


Motivation and Overview of Contribution

- Conditional random fields (CRFs) are a ubiquitous tool for **structured prediction**:
 - Allow the use of a high-dimensional feature set.
 - Formulated as convex optimization problem.
 - But **very slow to train**.
- Stochastic average gradient (SAG) methods are a new strategy for **convex optimization**:
 - Only look at a single training** example on each iteration, like stochastic gradient methods.
 - Linear convergence rate** similar to methods that process the entire dataset on every iteration.
- Our contribution is applying SAG with non-uniform sampling (NUS) to CRFs:
 - We **show how to reduce the memory requirements** using structure in the gradients.
 - We **propose a practical NUS scheme** that substantially improves empirical performance.
 - We **analyze the rate of convergence** of the SAGA variant under non-uniform sampling.
- SAG with NUS often outperforms existing methods for training CRFs.

Conditional Random Fields (CRFs)



- CRFs model probability of output $y \in \mathcal{Y}$ given input $x \in \mathcal{X}$ and features $F(x, y)$ using

$$p(y|x, w) = \frac{\exp(w^T F(x, y))}{\sum_{y'} \exp(w^T F(x, y'))}$$

- Given training examples $\{x_i, y_i\}$, standard approach is minimizing ℓ_2 -regularized NLL:

$$\min_w f(w) = \frac{1}{n} \sum_{i=1}^n -\log p(y_i|x_i, w) + \frac{\lambda}{2} \|w\|^2.$$

- Evaluating each $\log p(y_i|x_i, w)$ is expensive** due to sum over y' .

Related Work on Deterministic, Stochastic, and Hybrid Methods

- Deterministic gradient methods like L-BFGS [Wallach 2002, Sha & Pereira, 2003]:
 - Require $O(\log(1/\epsilon))$ iterations but n gradient evaluations per iteration.
- Stochastic gradient methods [Vishwanathan et al., 2006, Finkel et al., 2008]:
 - Require $O(1/\epsilon)$ iterations but only **1** gradient evaluation per iteration.
- Online exponentiated gradient [Collins et al., 2008]:
 - Requires $O(\log(1/\epsilon))$ iterations in terms of dual and **1** dual gradient evaluation per iteration.
- Hybrid deterministic-stochastic [Friedlander & Schmidt, 2012]:
 - Requires $O(\log(1/\epsilon))$ iterations and **growing** number of gradient evaluations per iteration.

Stochastic Average Gradient (SAG) for CRFs

- Stochastic average gradient [LeRoux, et al. 2012]:
 - Requires $O(\log(1/\epsilon))$ iterations and **1** gradient evaluation per iteration.
- SAG uses the iteration

$$w^{t+1} = w^t - \frac{\alpha}{n} \sum_{i=1}^n s_i^t,$$

where we set $s_i^t = -\nabla \log p(y_i|x_i, w^t) + \lambda w^t$ for one randomly-chosen training example.

- Challenge is the **memory** required for storing the s_i^t :

- $\nabla \log p(y_i|x_i, w^t)$ often sparse but **depends on number of features**
- λw^t is typically **dense**.

- Implementation issues for CRFs:

- Sparse trick 1**: to avoid storing λw^t use the exact gradient of the regularizer,

$$w^{t+1} = (1 - \alpha\lambda)w^t - \frac{\alpha}{n} \sum_{i=1}^n g_i^t,$$

where we set $g_i^t = -\nabla \log p(y_i|x_i, w^t)$ for one randomly-chosen example.

- Sparse trick 2**: use the representation $w^t = \beta^t v^t$ and 'lazy updates' to avoid dense vector operations.
- Step size**: we set $\alpha = 1/L$ with $L = L_g + \lambda$, and double approximation L_g when

$$f_i(w - (1/L)g_i) > f_i(w) - \frac{1}{2L_g} \|g_i\|^2, \quad (1)$$

but we multiply L_g by $2^{-1/n}$ after each iteration to **slowly increase step size**.

- Convergence**: we can stop if $\|\lambda w^t + \frac{1}{n} \sum_{i=1}^n g_i^t\|$ is sufficiently small.
- Reducing the memory**: for 'part-based' features, $F_j(x, y) = F(x) \mathbb{I}[y_k = s]$, the gradient has the form

$$\nabla_j \log p(y|x, w) = F(x) (\mathbb{I}[y_k = s] - p(y_k = s|x, w)).$$

and SAG update depends on differences in gradients,

$$\nabla_j \log p(y|x, w) - \nabla_j \log p(y|x, w_{old}) = F(x) (p(y_k = s|x, w_{old}) - p(y_k = s|x, w)),$$

so **we only need to store marginals $p(y_k = s|x, w)$** , that are **shared across features that depend on $y_k = s$** .

SAG with Practical Non-Uniform Sampling (NUS) Strategy

- Assume **each gradient has its own Lipschitz constant L_i** , a value such that

$$\|\nabla f_i(w) - \nabla f_i(v)\| \leq L_i \|w - v\|, \forall w, v,$$

bounding how fast gradient i can change.

- Key idea behind NUS**: bias sampling probability p_i towards Lipschitz constant L_i :

- Gradients that can change more quickly get updated more often.**
- Convergence rate depends on $\bar{L} = \text{mean}(L_i)$ instead of $L = \max(L_i)$.

- Practical 'partially-based' strategy:

- With probability 1/2 choose i uniformly.
- With probability 1/2 sample i with probability $L_i / \sum_j L_j$.
- Use a larger step-size of $\alpha = \frac{1}{2}(1/L + 1/\bar{L})$.
- Initialize with $L_i = \bar{L}$ the first time an example is chosen.
- Each time i is chosen, set $L_i = 0.9L_i$ then double it while (1) holds.
- If (1) holds ξ times (without backtracking), do not change L_i for $2^{\xi-1}$ next times i is sampled.
- Code: <http://www.cs.ubc.ca/~schmidtm/Software/SAG4CRF.html>.

Convergence Analysis for SAGA with Non-Uniform Sampling

- We **analyze a NUS extension of SAGA**, which has similar performance but easier analysis.

- Let the sequences $\{w^t\}$ and $\{s_j^t\}$ be defined by

$$w^{t+1} = w^t - \alpha \left[\frac{1}{np_j} (\nabla f_{j_t}(w^t) - s_{j_t}^t) + \frac{1}{n} \sum_{i=1}^n s_i^t \right],$$

$$s_j^{t+1} = \begin{cases} \nabla f_{j_t}(w^t) & \text{if } j = r_t, \\ s_j^t & \text{otherwise.} \end{cases}$$

where j_t is chosen with probability p_j .

- (a) If r_t is set to j_t , then with $\alpha = \frac{n\rho_{\min}}{4L+n\mu}$ we have

$$\mathbb{E}[\|w^t - w^*\|^2] \leq (1 - \mu\alpha)^t [\|x^0 - x^*\| + C_a],$$

where $\rho_{\min} = \min_i \{p_i\}$ and

$$C_a = \frac{2\rho_{\min}}{(4L+n\mu)^2} \sum_{i=1}^n \frac{1}{p_i} \|\nabla f_i(x^0) - \nabla f_i(x^*)\|^2.$$

- (b) If $p_j = \frac{L_j}{\sum_{i=1}^n L_i}$ and r_t is chosen uniformly at random, then with $\alpha = \frac{1}{4L}$ we have

$$\mathbb{E}[\|w^t - w^*\|^2] \leq \left(1 - \min\left\{\frac{1}{3n}, \frac{\mu}{8L}\right\}\right)^t [\|x^0 - x^*\| + C_b],$$

where:

$$C_b = \frac{n}{2\bar{L}} [f(x^0) - f(x^*)]$$

- (a) **SAGA has a linear convergence rate whenever $p_i > 0$ for all i .**

- (b) **SAGA has a faster rate with p_i proportional to L_i and generating a uniform sample.**

Experiment Results

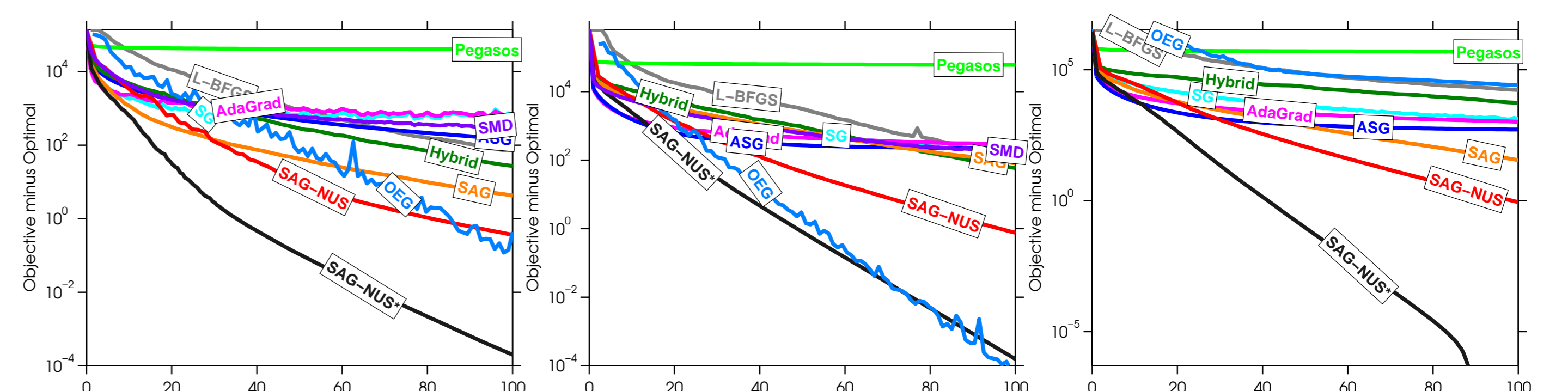


Figure: Training objective sub-optimality against effective number of passes for OCR, CONLL-2000, POS.

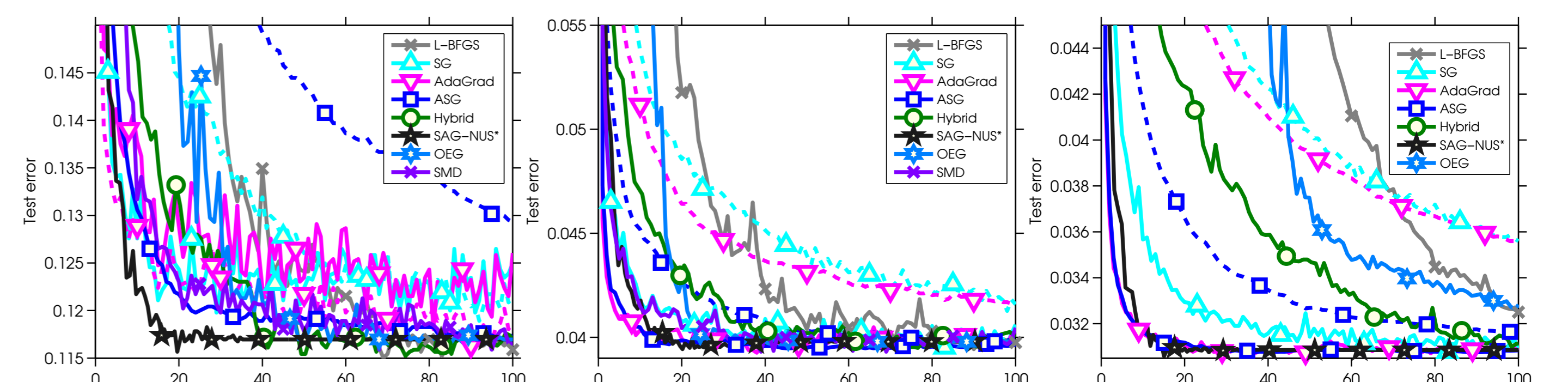


Figure: Test error against effective number of passes, for OCR, CONLL-2000, POS. (Dashed lines are stochastic method with sub-optimal step-size.)

Discussion

- If memory requirements prohibitive, use mini-batches or SVRG [Johnson & Zhang, 2013].
- Could use ℓ_1 -regularization with proximal versions [Defazio et al., 2014].
- Algorithms applies to any graph structure and approximate inference could be used.
- Could use multi-core computation and distributed parallel implementations.