Non-Uniform Stochastic Average Gradient for Training Conditional Random Fields

Mark Schmidt, Reza Babanezhad, Mohamed Ahmed
Ann Clifton, Anoop Sarkar

University of British Columbia, Simon Fraser University

NIPS Optimization Workshop, 2014

Motivation: Structured Prediction

Classical supervised learning:

Input: 

Output: "P"

Motivation: Structured Prediction

Classical supervised learning:

Input: 

Output: "P"

Structured prediction:

Input: 

Output: "Paris"

Motivation: Structured Prediction

Classical supervised learning:

Input: 

Output: "P"

Structured prediction:

Input: 

Output: "Paris"

Other structure prediction tasks:

- Labelling all people/places in Wikiepdia, finding coding regions in DNA sequences, labelling all voxels in an MRI as normal or tumor, predicting protein structure from sequence, weather forecasting, translating from French to English,etc.

Motivation: Structured Prediction

Naive approaches to predicting letters y given images x:

- Multinomial logistic regression to predict word:

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

Motivation: Structured Prediction

Naive approaches to predicting letters y given images x:

- Multinomial logistic regression to predict word:

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

This requires parameter vector w_k for all possible words k.

Motivation: Structured Prediction

Naive approaches to predicting letters y given images x:

- Multinomial logistic regression to predict word:

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

  This requires parameter vector w_k for all possible words k.

- Multinomial logistic regression to predict each letter:

$$p(y_j|x_j, w) = \frac{\exp(w_{y_j}^T F(x_j))}{\sum_{y_j'} \exp(w_{y_j'}^T F(x_j))}.$$

  This works if you are really good at predicting individual letters.

Motivation: Structured Prediction

Naive approaches to predicting letters y given images x:

- Multinomial logistic regression to predict word:

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

  This requires parameter vector w_k for all possible words k.

- Multinomial logistic regression to predict each letter:

$$p(y_j|x_j, w) = \frac{\exp(w_{y_j}^T F(x_j))}{\sum_{y_j'} \exp(w_{y_j'}^T F(x_j))}.$$

  This works if you are really good at predicting individual letters.

  But this ignores dependencies between letters.

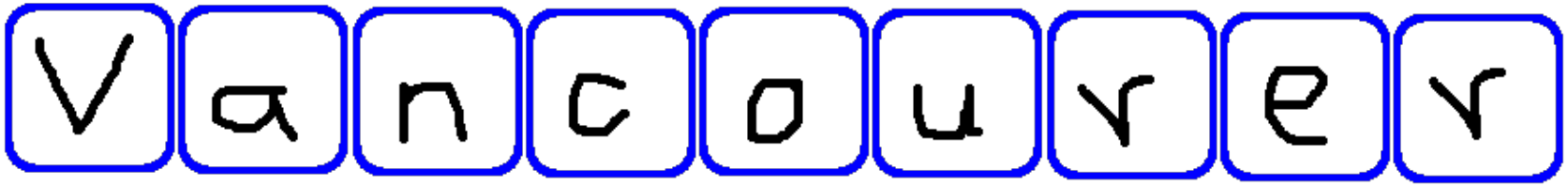
Motivation: Structured Prediction

- What letter is this?

Motivation: Structured Prediction

- What letter is this?

- What are these letters?



Conditional Random Fields

- Conditional random fields model targets y given inputs x using

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

where w are the parameters.

Conditional Random Fields

- Conditional random fields model targets y given inputs x using

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

  where w are the parameters.
- Examples of features $F(y,x)$:
  - $F(y_j,x)$: these features lead to a logistic model for each letter.

Conditional Random Fields

- Conditional random fields model targets y given inputs x using

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

  where w are the parameters.
- Examples of features $F(y,x)$:
  - $F(y_j, x)$: these features lead to a logistic model for each letter.
  - $F(y_{j-1}, y_j, x)$: dependency between adjacent letters ('q-u').

Conditional Random Fields

- Conditional random fields model targets y given inputs x using

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

  where w are the parameters.
- Examples of features $F(y,x)$:
  - $F(y_j, x)$: these features lead to a logistic model for each letter.
  - $F(y_{j-1}, y_j, x)$: dependency between adjacent letters ('q-u').
  - $F(y_{j-1}, y_j, j, x)$: position-based dependency (French: 'e-r' ending).

Conditional Random Fields

- Conditional random fields model targets y given inputs x using

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

  where w are the parameters.
- Examples of features $F(y,x)$:
  - $F(y_j, x)$: these features lead to a logistic model for each letter.
  - $F(y_{j-1}, y_j, x)$: dependency between adjacent letters ('q-u').
  - $F(y_{j-1}, y_j, j, x)$: position-based dependency (French: 'e-r' ending).
  - $F(y_{j-2}, y_{j-1}, y_j, j, x)$: third-order and position (English: 'i-n-g' end).

Conditional Random Fields

- Conditional random fields model targets y given inputs x using

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

  where w are the parameters.
- Examples of features $F(y,x)$:
  - $F(y_j, x)$: these features lead to a logistic model for each letter.
  - $F(y_{j-1}, y_j, x)$: dependency between adjacent letters ('q-u').
  - $F(y_{j-1}, y_j, j, x)$: position-based dependency (French: 'e-r' ending).
  - $F(y_{j-2}, y_{j-1}, y_j, j, x)$: third-order and position (English: 'i-n-g' end).
  - $F(y \in \mathcal{D}, x)$: is y in dictionary $\mathcal{D}$?

Conditional Random Fields

- Conditional random fields model targets y given inputs x using

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

  where w are the parameters.
- Examples of features $F(y,x)$:
  - $F(y_j, x)$: these features lead to a logistic model for each letter.
  - $F(y_{j-1}, y_j, x)$: dependency between adjacent letters ('q-u').
  - $F(y_{j-1}, y_j, j, x)$: position-based dependency (French: 'e-r' ending).
  - $F(y_{j-2}, y_{j-1}, y_j, j, x)$: third-order and position (English: 'i-n-g' end).
  - $F(y \in \mathcal{D}, x)$: is y in dictionary $\mathcal{D}$?
- CRFs are a ubiquitous tool in natural language processing:
  - Part-of-speech tagging, semantic role labelling, information extraction, shallow parsing, named-entity recognition, etc.

Optimization Formulation and Challenge

- Typically train using ℓ_2-regularized negative log-likelihood:

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

Optimization Formulation and Challenge

- Typically train using ℓ_2-regularized negative log-likelihood:

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

- Good news: $\nabla f(w)$ is Lipschitz-continuous, f is strongly-convex.

Optimization Formulation and Challenge

- Typically train using ℓ_2-regularized negative log-likelihood:

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

- Good news: $\nabla f(w)$ is Lipschitz-continuous, f is strongly-convex.
- Bad news: evaluating $\log p(y_i|x_i, w)$ and its gradient is expensive.

Optimization Formulation and Challenge

- Typically train using ℓ_2-regularized negative log-likelihood:

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

- Good news: $\nabla f(w)$ is Lipschitz-continuous, f is strongly-convex.
- Bad news: evaluating $\log p(y_i|x_i, w)$ and its gradient is expensive.
  - Chain-structures: run forward-backward on each example.

Optimization Formulation and Challenge

- Typically train using ℓ_2-regularized negative log-likelihood:

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

- Good news: $\nabla f(w)$ is Lipschitz-continuous, f is strongly-convex.
- Bad news: evaluating $\log p(y_i|x_i, w)$ and its gradient is expensive.
  - Chain-structures: run forward-backward on each example.
  - General features: exponential in tree-width of dependency graph.
  - A lot of work on approximate evaluation.
- This optimization problem remains a bottleneck.

Current Optimization Methods

- Lafferty et al. [2001] proposed an iterative scaling approach.
- Outperformed by L-BFGS quasi-Newton algorithm.

[Wallach, 2002, Sha Pereira, 2003]

  - Has a linear convergence rate: $O(\log(1/\epsilon))$ iterations required.

Current Optimization Methods

- Lafferty et al. [2001] proposed an iterative scaling approach.
- Outperformed by L-BFGS quasi-Newton algorithm.

[Wallach, 2002, Sha Pereira, 2003]

  - Has a linear convergence rate: $O(\log(1/\epsilon))$ iterations required.
  - But each iteration requires $\log p(y_i|x_i, w)$ for all n examples.

Current Optimization Methods

- Lafferty et al. [2001] proposed an iterative scaling approach.
- Outperformed by L-BFGS quasi-Newton algorithm.

  [Wallach, 2002, Sha Pereira, 2003]

  - Has a linear convergence rate: $O(\log(1/\epsilon))$ iterations required.
  - But each iteration requires $\log p(y_i|x_i, w)$ for all n examples.
- To scale to large n, stochastic gradient methods were examined.

  [Vishwanathan et al., 2006]

  - Iteration cost is independent of n.

Current Optimization Methods

- Lafferty et al. [2001] proposed an iterative scaling approach.
- Outperformed by L-BFGS quasi-Newton algorithm.

[Wallach, 2002, Sha Pereira, 2003]

  - Has a linear convergence rate: $O(\log(1/\epsilon))$ iterations required.
  - But each iteration requires $\log p(y_i|x_i, w)$ for all n examples.
- To scale to large n, stochastic gradient methods were examined.

[Vishwanathan et al., 2006]

  - Iteration cost is independent of n.
  - But has a sub linear convergence rate: $O(1/\epsilon)$ iterations required.
  - Or with constant step-size you get linear rate up to fixed tolerance.

Current Optimization Methods

- Lafferty et al. [2001] proposed an iterative scaling approach.
- Outperformed by L-BFGS quasi-Newton algorithm.

  [Wallach, 2002, Sha Pereira, 2003]

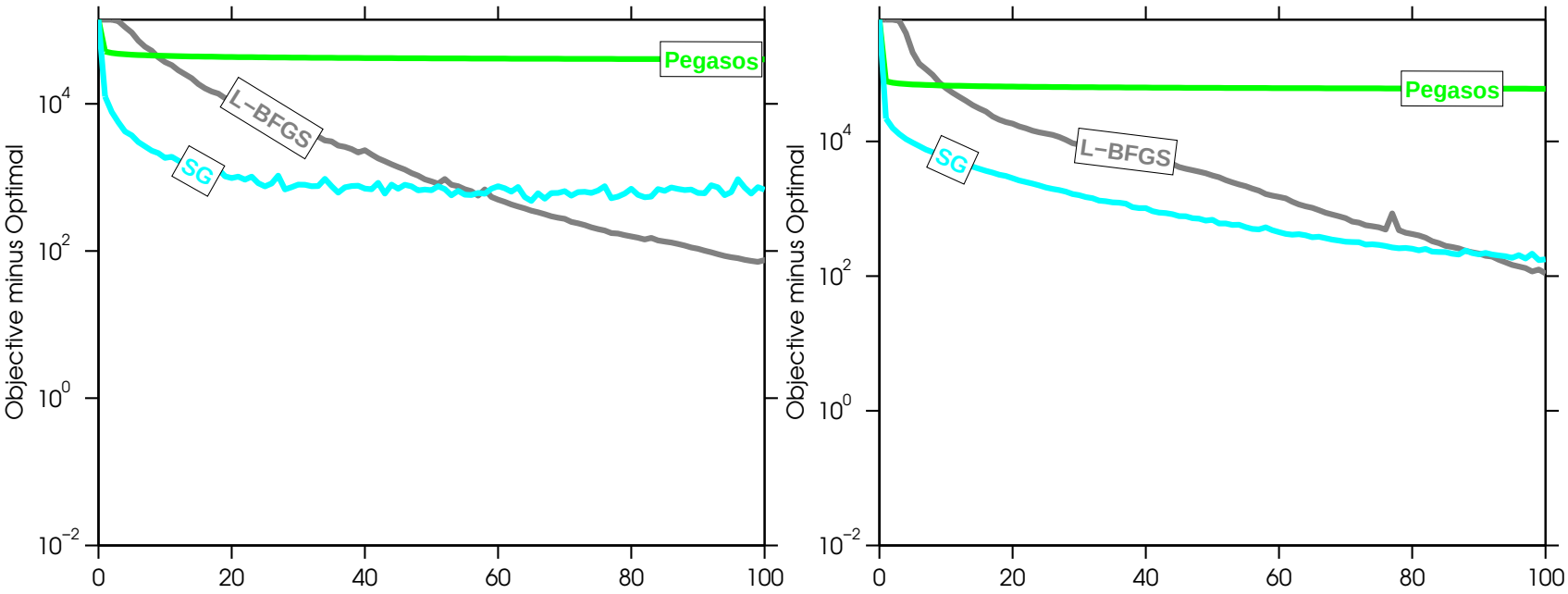
  - Has a linear convergence rate: $O(\log(1/\epsilon))$ iterations required.
  - But each iteration requires $\log p(y_i|x_i, w)$ for all n examples.
- To scale to large n, stochastic gradient methods were examined.

  [Vishwanathan et al., 2006]

  - Iteration cost is independent of n.
  - But has a sub linear convergence rate: $O(1/\epsilon)$ iterations required.
  - Or with constant step-size you get linear rate up to fixed tolerance.
- These remain the strategies used by most implementations.
  - Many packages implement both strategies.

L-BFGS vs. Stochastic Gradient

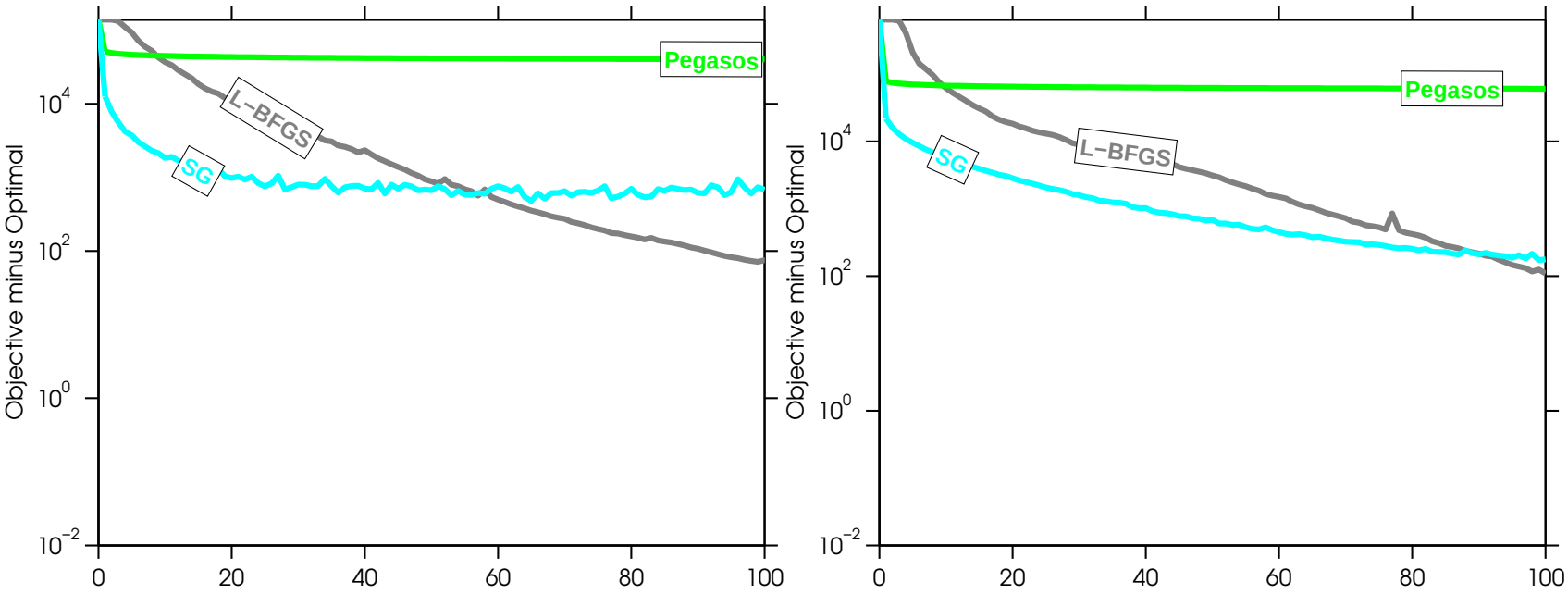
- L-BFGS has fast convergence but slow iterations.
- SG (decreasing α) has slow convergence but fast iterations.
- SG (constant α) has fast convergence but not to optimal.



(Using $\alpha_t = \alpha/(\delta + \sqrt{t})$ gives intermediate performance.)

L-BFGS vs. Stochastic Gradient

- L-BFGS has fast convergence but slow iterations.
- SG (decreasing α) has slow convergence but fast iterations.
- SG (constant α) has fast convergence but not to optimal.



(Using $\alpha_t = \alpha/(\delta + \sqrt{t})$ gives intermediate performance.)

- Can we develop a method that outperforms these methods?

Attempts to speed up CRF training

- Averaged stochastic gradient with large step-sizes (ASG):

  [Polyak & Juditsky, 1992, Bach & Moulines, 2011]

  - Tends to outperform non-averaged SG.
  - Can be outperformed by L-BFGS.

Attempts to speed up CRF training

- Averaged stochastic gradient with large step-sizes (ASG):

  [Polyak & Juditsky, 1992, Bach & Moulines, 2011]

  - Tends to outperform non-averaged SG.
  - Can be outperformed by L-BFGS.
- Adaptive diagonal scaling (AdaGrad): [Duchi et al., 2010]
  - Improved regret bounds but still $O(1/\epsilon)$ rate.
  - Often improves performance over basic stochastic gradient.

Attempts to speed up CRF training

- Averaged stochastic gradient with large step-sizes (ASG):

  [Polyak & Juditsky, 1992, Bach & Moulines, 2011]

  - Tends to outperform non-averaged SG.
  - Can be outperformed by L-BFGS.
- Adaptive diagonal scaling (AdaGrad): [Duchi et al., 2010]
  - Improved regret bounds but still $O(1/\epsilon)$ rate.
  - Often improves performance over basic stochastic gradient.
  - Often outperformed by ASG.

Attempts to speed up CRF training

- Averaged stochastic gradient with large step-sizes (ASG):

  [Polyak & Juditsky, 1992, Bach & Moulines, 2011]

  - Tends to outperform non-averaged SG.
  - Can be outperformed by L-BFGS.
- Adaptive diagonal scaling (AdaGrad): [Duchi et al., 2010]
  - Improved regret bounds but still $O(1/\epsilon)$ rate.
  - Often improves performance over basic stochastic gradient.
  - Often outperformed by ASG.
- Hybrid of L-BFGS and stochastic gradient: [Frielander & Schmidt, 2012]
  - $O(\log(1/\epsilon))$ rate but cheaper in early iterations.
  - Improved performance over L-BFGS.

Attempts to speed up CRF training

- Averaged stochastic gradient with large step-sizes (ASG):

  [Polyak & Juditsky, 1992, Bach & Moulines, 2011]

  - Tends to outperform non-averaged SG.
  - Can be outperformed by L-BFGS.
- Adaptive diagonal scaling (AdaGrad): [Duchi et al., 2010]
  - Improved regret bounds but still $O(1/\epsilon)$ rate.
  - Often improves performance over basic stochastic gradient.
  - Often outperformed by ASG.
- Hybrid of L-BFGS and stochastic gradient: [Frielander & Schmidt, 2012]
  - $O(\log(1/\epsilon))$ rate but cheaper in early iterations.
  - Improved performance over L-BFGS.
  - Sometimes better and sometimes worse than ASG.

Attempts to speed up CRF training

- Averaged stochastic gradient with large step-sizes (ASG):

  [Polyak & Juditsky, 1992, Bach & Moulines, 2011]

  - Tends to outperform non-averaged SG.
  - Can be outperformed by L-BFGS.
- Adaptive diagonal scaling (AdaGrad): [Duchi et al., 2010]
  - Improved regret bounds but still $O(1/\epsilon)$ rate.
  - Often improves performance over basic stochastic gradient.
  - Often outperformed by ASG.
- Hybrid of L-BFGS and stochastic gradient: [Frielander & Schmidt, 2012]
  - $O(\log(1/\epsilon))$ rate but cheaper in early iterations.
  - Improved performance over L-BFGS.
  - Sometimes better and sometimes worse than ASG.
- Stochastic dual block-coordinate exponentiated gradient ascent:

  [Collin et al., 2008]

  - $O(log(1/\epsilon))$ iterations for dual problem with $O(1)$ cost.
  - In theory, the rate of deterministic with the cost of stochastic.

Attempts to speed up CRF training

- Averaged stochastic gradient with large step-sizes (ASG):

  [Polyak & Juditsky, 1992, Bach & Moulines, 2011]

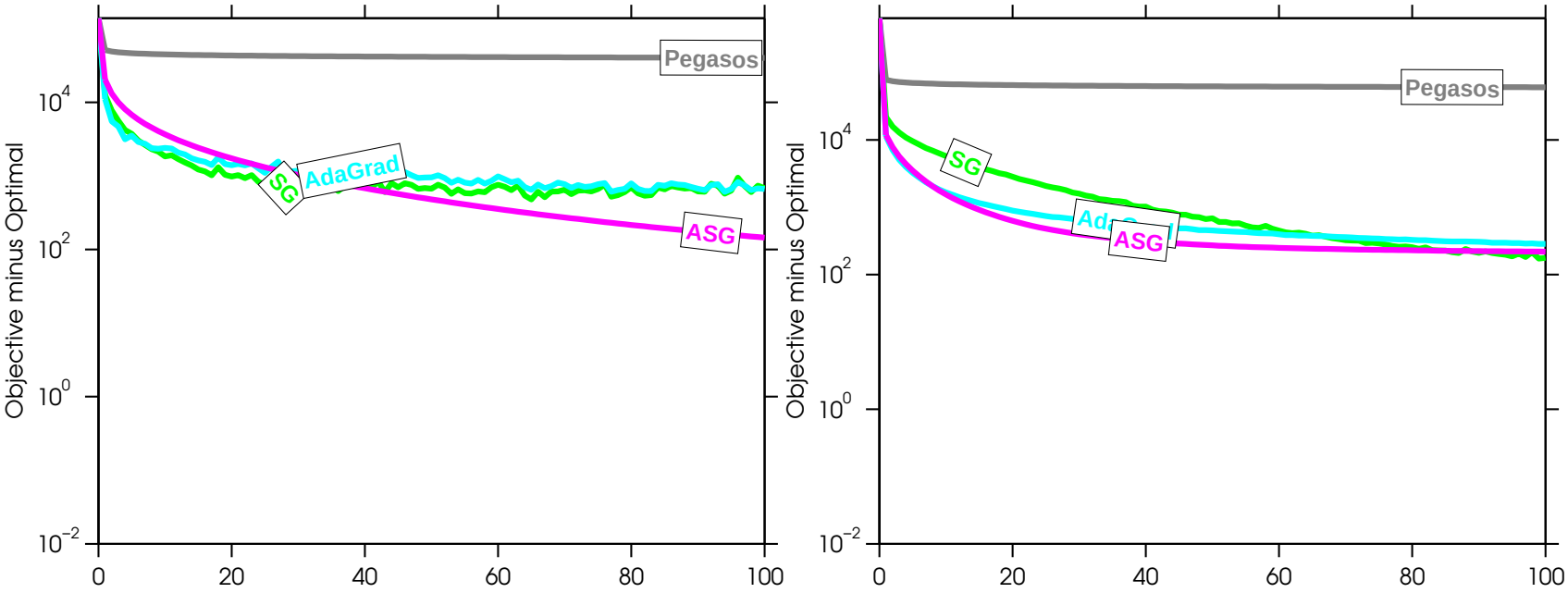
  - Tends to outperform non-averaged SG.
  - Can be outperformed by L-BFGS.
- Adaptive diagonal scaling (AdaGrad): [Duchi et al., 2010]
  - Improved regret bounds but still $O(1/\epsilon)$ rate.
  - Often improves performance over basic stochastic gradient.
  - Often outperformed by ASG.
- Hybrid of L-BFGS and stochastic gradient: [Frielander & Schmidt, 2012]
  - $O(\log(1/\epsilon))$ rate but cheaper in early iterations.
  - Improved performance over L-BFGS.
  - Sometimes better and sometimes worse than ASG.
- Stochastic dual block-coordinate exponentiated gradient ascent:

  [Collin et al., 2008]

  - $O(log(1/\epsilon))$ iterations for dual problem with $O(1)$ cost.
  - In theory, the rate of deterministic with the cost of stochastic.
  - Often gives poor performance with small λ.

Comparison of Stochastic Gradient Methods

- Comparison of Pegasos, SG, ASG, and AdaGrad:

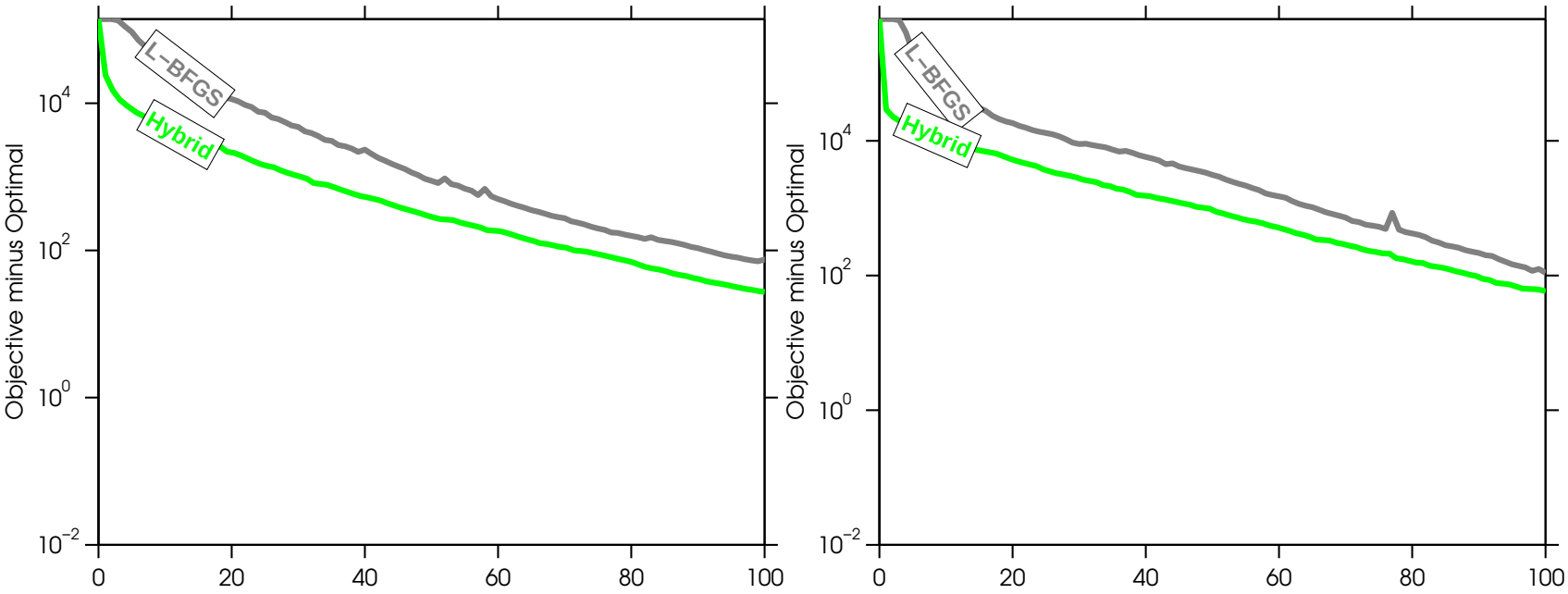


(Averaging did not improve performance of Pegasos.)

- ASG often outperforms SG and AdaGrad.

Comparison of L-BFGS Methods

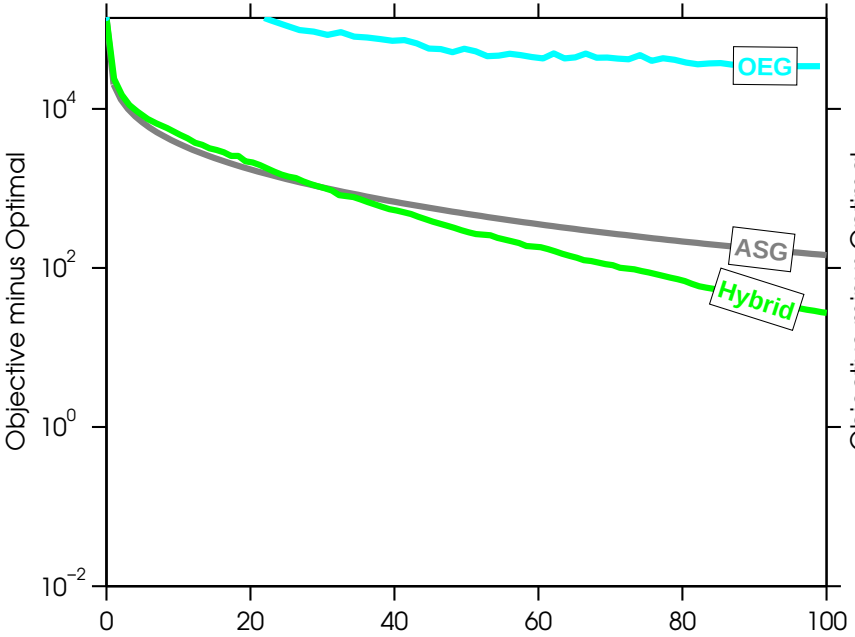
- Comparison of L-BFGS and Hybrid Stochastic/L-BFGS:



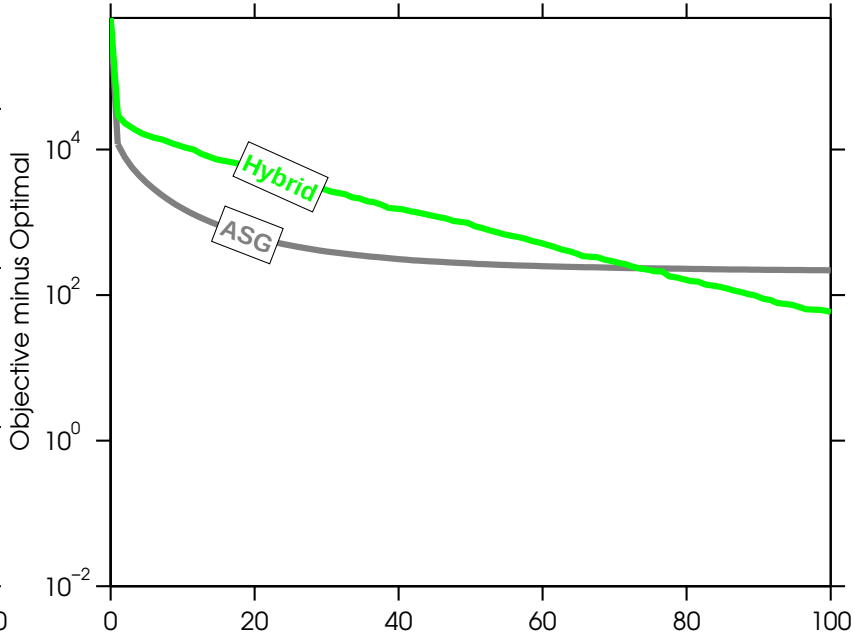
- Hybrid often outperforms L-BFGS.

Comparison with dual exponentiated gradient

- Comparison of ASG, Hybrid, and OEG:



(EG performs better if λ is small.)

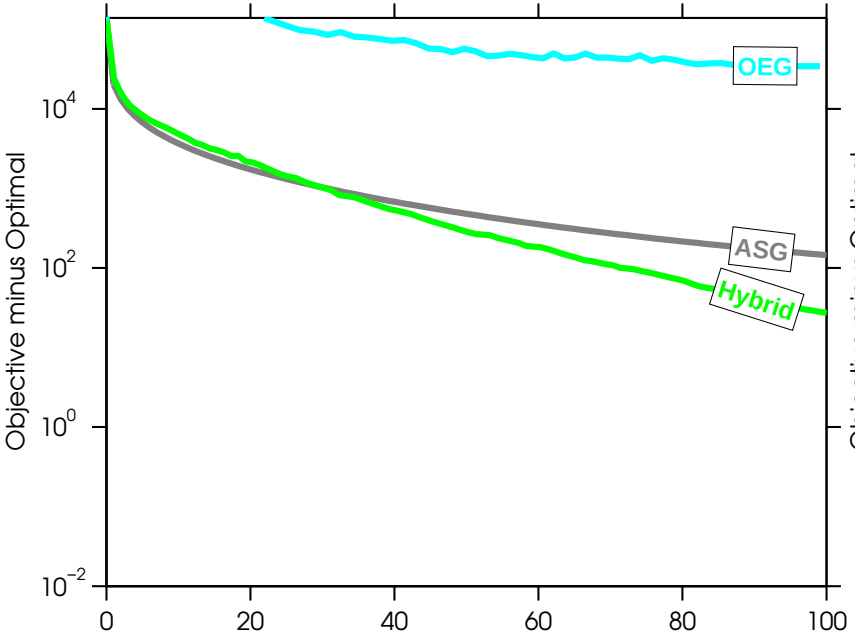


(OEG was not run on this data).

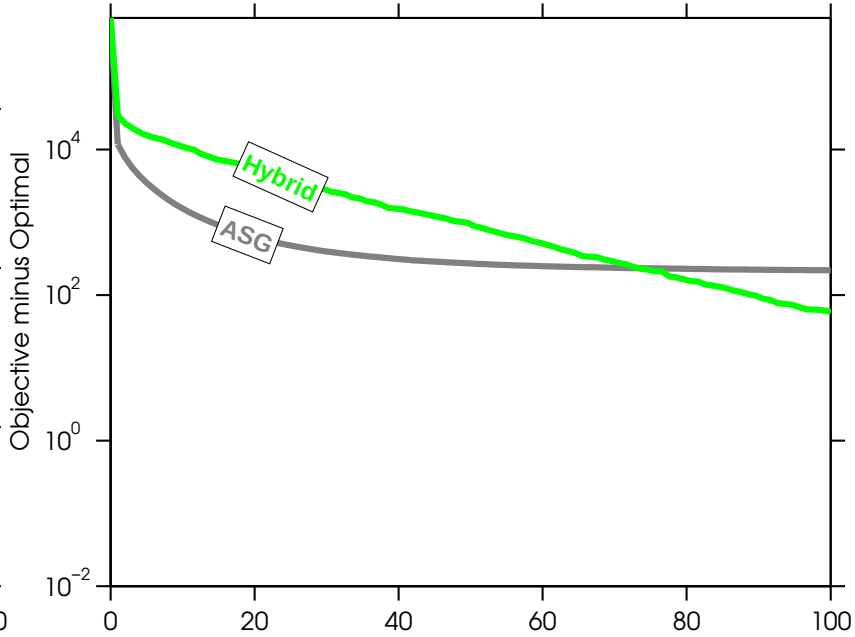
- OEG is worse than other competitive methods.
- Hybrid vs. ASG is problem-dependent.

Comparison with dual exponentiated gradient

- Comparison of ASG, Hybrid, and OEG:



(EG performs better if λ is small.)



(OEG was not run on this data).

- OEG is worse than other competitive methods.
- Hybrid vs. ASG is problem-dependent.
- Fancier methods do not give consistent/significant improvement.

A New Hope: Linearly-Convergent Stochastic Gradient

- Recent new stochastic algorithms for minimizing finite sums,

$$\min_w f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(x),$$

requiring $O(\log(1/\epsilon))$ iterations with $O(1)$ cost.

A New Hope: Linearly-Convergent Stochastic Gradient

- Recent new stochastic algorithms for minimizing finite sums,

$$\min_w f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(x),$$

requiring $O(\log(1/\epsilon))$ iterations with $O(1)$ cost.

- Stochastic average gradient (SAG): [Le Roux et al., 2012]

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

where iteration sets $s_i^t = \nabla f_i(x^t)$ for random i (o.w., $s_i^t = s_i^{t-1}$).

A New Hope: Linearly-Convergent Stochastic Gradient

- Recent new stochastic algorithms for minimizing finite sums,

$$\min_w f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(x),$$

requiring $O(\log(1/\epsilon))$ iterations with $O(1)$ cost.

- Stochastic average gradient (SAG): [Le Roux et al., 2012]

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

where iteration sets $s_i^t = \nabla f_i(x^t)$ for random i (o.w., $s_i^t = s_i^{t-1}$).

- Similar rate to full gradient but iterations are n times cheaper.

A New Hope: Linearly-Convergent Stochastic Gradient

- Recent new stochastic algorithms for minimizing finite sums,

$$\min_w f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(x),$$

requiring $O(\log(1/\epsilon))$ iterations with $O(1)$ cost.

- Stochastic average gradient (SAG): [Le Roux et al., 2012]

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

where iteration sets $s_i^t = \nabla f_i(x^t)$ for random i (o.w., $s_i^t = s_i^{t-1}$).

- Similar rate to full gradient but iterations are n times cheaper.
- Unlike EG, adaptive to strong-convexity.

Comparison of Convergence Rates

Number of iterations to reach an accuracy of ϵ:

Deterministic:	$O(n\sqrt{\frac{L}{\mu}}\log(1/\epsilon))$	(primal)
Stochastic	$O(\frac{\sigma^2}{\mu\epsilon} + \sqrt{\frac{L}{\mu}}\log(1/\epsilon))$	(primal)
Dual stochastic EG	$O((n + \frac{L}{\lambda})\log(1/\epsilon))$	(dual)
SAG	$O((n + \frac{L}{\mu})\log(1/\epsilon))$	(primal)

Comparison of Convergence Rates

Number of iterations to reach an accuracy of ϵ:

Deterministic:	$O(n\sqrt{\frac{L}{\mu}}\log(1/\epsilon))$	(primal)
Stochastic	$O(\frac{\sigma^2}{\mu\epsilon} + \sqrt{\frac{L}{\mu}}\log(1/\epsilon))$	(primal)
Dual stochastic EG	$O((n + \frac{L}{\lambda})\log(1/\epsilon))$	(dual)
SAG	$O((n + \frac{L}{\mu})\log(1/\epsilon))$	(primal)

Similar to deterministic methods, SAG can adapt to problem:

- SAG automatically adapts to local μ at solution.
- Practical implementations try to automatically adapt to L, too.

Strong empirical performance for independent classification.

Addressing the Memory Requirements

- Could this algorithm consistently outperform the old methods?

Addressing the Memory Requirements

- Could this algorithm consistently outperform the old methods?
- First, we need to address that SAG requires storing n gradients,

$$s_i^t = \lambda w^k - \nabla \log p(y_i | x_i, w^k),$$

for some previous k, which do not have a nice structure.

Addressing the Memory Requirements

- Could this algorithm consistently outperform the old methods?
- First, we need to address that SAG requires storing n gradients,

$$s_i^t = \lambda w^k - \nabla \log p(y_i | x_i, w^k),$$

  for some previous k, which do not have a nice structure.
- We could use SVRG/mixedGrad:

  [Johnson & Zhang, 2013, Mahdavi et al, 2013,]

  - Similar convergence rate but without memory requirement.

Addressing the Memory Requirements

- Could this algorithm consistently outperform the old methods?
- First, we need to address that SAG requires storing n gradients,

$$s_i^t = \lambda w^k - \nabla \log p(y_i | x_i, w^k),$$

  for some previous k, which do not have a nice structure.
- We could use SVRG/mixedGrad:

  [Johnson & Zhang, 2013, Mahdavi et al, 2013,]

  - Similar convergence rate but without memory requirement.
  - But requires two evaluations of $\nabla \log p(y_i | x_i, w^t)$ per iteration.

Addressing the Memory Requirements

- The deterministic gradient update can be written:

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

Addressing the Memory Requirements

- The deterministic gradient update can be written:

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

- The SAG update:

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

where $s_i^t = \lambda w^k - \nabla \log p(y_i|x_i, w^k)$ for some previous k.

Addressing the Memory Requirements

- The deterministic gradient update can be written:

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

- The SAG update:

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

where $s_i^t = \lambda w^k - \nabla \log p(y_i|x_i, w^k)$ for some previous k.

- A modified update where we don't approximate the regularizer:

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

where $g_i^t = -\nabla \log p(y_i|x_i, w^k)$ for some previous k.

Addressing the Memory Requirements

- The deterministic gradient update can be written:

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

- The SAG update:

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

where $s_i^t = \lambda w^k - \nabla \log p(y_i|x_i, w^k)$ for some previous k.

- A modified update where we don't approximate the regularizer:

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

where $g_i^t = -\nabla \log p(y_i|x_i, w^k)$ for some previous k.

- The g_i^t have a nice structure, and regularizer update is efficient.

Addressing the Memory Requirements

- Consider a chain-structured CRF model of the form

$$p(y|x, w) \propto \exp\left(\sum_{j=1}^{V} x_j^T w_{y_j} + \sum_{j=1}^{V-1} w_{y_j, y_{j+1}}\right).$$

Addressing the Memory Requirements

- Consider a chain-structured CRF model of the form

$$p(y|x, w) \propto \exp\left(\sum_{j=1}^{V} x_j^T w_{y_j} + \sum_{j=1}^{V-1} w_{y_j, y_{j+1}}\right).$$

- The gradient with respect to a particular vector w_k is

$$\nabla_{w_k} \log\ p(y|x, w) = \sum_{j=1}^{V} x_j \left[\mathbb{I}(y_j = k) - p(y_j = k|x, w)\right].$$

Addressing the Memory Requirements

- Consider a chain-structured CRF model of the form

$$p(y|x,w) \propto \exp\left(\sum_{j=1}^{V} x_j^T w_{y_j} + \sum_{j=1}^{V-1} w_{y_j,y_{j+1}}\right).$$

- The gradient with respect to a particular vector w_k is

$$\nabla_{w_k} \log\ p(y|x,w) = \sum_{j=1}^{V} x_j \left[\mathbb{I}(y_j = k) - p(y_j = k|x,w)\right].$$

- The modified SAG algorithm needs to update the sum,

$$\sum_{i=1}^{n} g_i^{t+1} = \sum_{i=1}^{n} [g_i^t] + g_i^{t+1} - g_i^t.$$

Addressing the Memory Requirements

- Consider a chain-structured CRF model of the form

$$p(y|x,w) \propto \exp\left(\sum_{j=1}^{V} x_j^T w_{y_j} + \sum_{j=1}^{V-1} w_{y_j,y_{j+1}}\right).$$

- The gradient with respect to a particular vector w_k is

$$\nabla_{w_k} \log\ p(y|x,w) = \sum_{j=1}^{V} x_j \left[\mathbb{I}(y_j = k) - p(y_j = k|x,w)\right].$$

- The modified SAG algorithm needs to update the sum,

$$\sum_{i=1}^{n} g_i^{t+1} = \sum_{i=1}^{n} [g_i^t] + g_i^{t+1} - g_i^t.$$

- To do this, we only need to store the unary marginals.

Addressing the Memory Requirements

- Consider a chain-structured CRF model of the form

$$p(y|x,w) \propto \exp\left(\sum_{j=1}^{V} x_j^T w_{y_j} + \sum_{j=1}^{V-1} w_{y_j,y_{j+1}}\right).$$

- The gradient with respect to a particular vector w_k is

$$\nabla_{w_k} \log\ p(y|x,w) = \sum_{j=1}^{V} x_j \left[\mathbb{I}(y_j = k) - p(y_j = k|x,w)\right].$$

- The modified SAG algorithm needs to update the sum,

$$\sum_{i=1}^{n} g_i^{t+1} = \sum_{i=1}^{n} [g_i^t] + g_i^{t+1} - g_i^t.$$

- To do this, we only need to store the unary marginals.
- General pairwise graphical models require $O(VK + EK^2)$.
- Unlike basic SAG, no dependence on number of features.

Practical issues: setting the step size and stopping

Traditional sources of frustration for stochastic gradient users:

1. Need to choose between slow convergence or oscillations.
2. Setting the sequence of step-sizes.
3. Deciding when to stop.

Practical issues: setting the step size and stopping

Traditional sources of frustration for stochastic gradient users:

1. Need to choose between slow convergence or oscillations.
2. Setting the sequence of step-sizes.
3. Deciding when to stop.

These are easier to address in methods like SAG:

1. Faster convergence rates.
2. Allow a constant step-size ($\alpha = 1/L$).
3. Approximate the full gradient for deciding when to stop.

Practical issues: setting the step size and stopping

No manual step-size tuning, we approximate L as we go:

- Start with $L = 1$.

Practical issues: setting the step size and stopping

No manual step-size tuning, we approximate L as we go:

- Start with $L = 1$.
- If $\|f_i'(x)\|^2 \geq \delta$, increase L until we satisfy:

$$f_i(x - \frac{1}{L}f_i'(x)) \leq f_i'(x) - \frac{1}{2L}\|f_i'(x)\|^2.$$

(Lipschitz approximation procedure from FISTA)

Practical issues: setting the step size and stopping

No manual step-size tuning, we approximate L as we go:

- Start with $L = 1$.
- If $\|f_i'(x)\|^2 \geq \delta$, increase L until we satisfy:

$$f_i(x - \frac{1}{L}f_i'(x)) \leq f_i'(x) - \frac{1}{2L}\|f_i'(x)\|^2.$$

  (Lipschitz approximation procedure from FISTA)
- Decrease L between iterations.
  (makes algorithm adaptive to local L)

Practical issues: setting the step size and stopping

No manual step-size tuning, we approximate L as we go:

- Start with $L = 1$.
- If $\|f_i'(x)\|^2 \geq \delta$, increase L until we satisfy:

$$f_i(x - \frac{1}{L}f_i'(x)) \leq f_i'(x) - \frac{1}{2L}\|f_i'(x)\|^2.$$

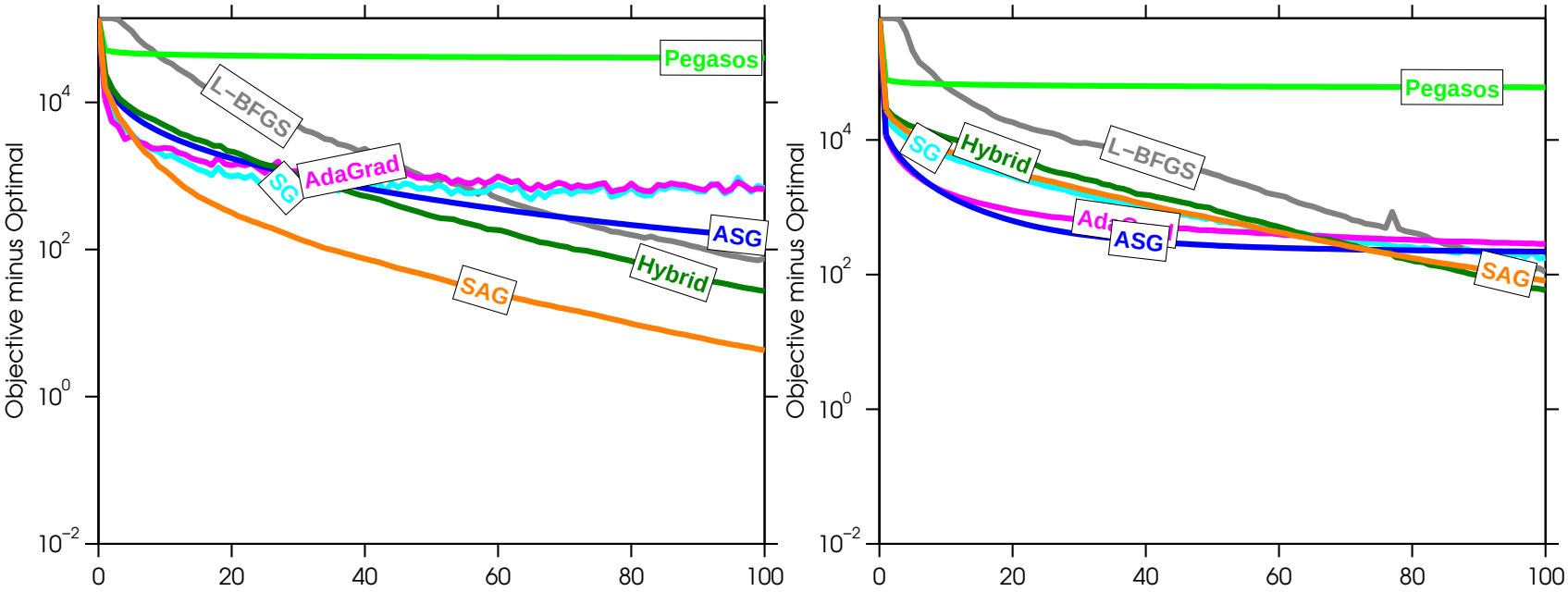
  (Lipschitz approximation procedure from FISTA)
- Decrease L between iterations.

  (makes algorithm adaptive to local L)

Performs similar to choosing the optimal step-size.

Comparison of SAG to existing methods

- Comparison of SAG and state of the art methods.



- Sometimes better and sometimes worse than existing methods.
- Have we really made so little progress???

Non-Uniform Sampling

- Recent works examining non-uniform sampling (NUS):
  - Cyclic projection [Strohmer & Vershynin, 2009].
  - Coordinate descent [Nesterov, 2010].
  - SAG [Schmidt et al, 2013], heuristic argument/experiments.
  - SVRG [Xiao & Zhang, 2014].
  - Stochastic gradient [Needell et al., 2014].
- Appropriate NUS yields faster convergence rates.

Non-Uniform Sampling

- Recent works examining non-uniform sampling (NUS):
  - Cyclic projection [Strohmer & Vershynin, 2009].
  - Coordinate descent [Nesterov, 2010].
  - SAG [Schmidt et al, 2013], heuristic argument/experiments.
  - SVRG [Xiao & Zhang, 2014].
  - Stochastic gradient [Needell et al., 2014].
- Appropriate NUS yields faster convergence rates.
- Key idea: bias sampling towards Lipschitz constants.
  - "If a gradient can change quickly, sample it more often".
  - "If a gradient can only change slowly, don't sample if often".

Non-Uniform Sampling

- Recent works examining non-uniform sampling (NUS):
  - Cyclic projection [Strohmer & Vershynin, 2009].
  - Coordinate descent [Nesterov, 2010].
  - SAG [Schmidt et al, 2013], heuristic argument/experiments.
  - SVRG [Xiao & Zhang, 2014].
  - Stochastic gradient [Needell et al., 2014].
- Appropriate NUS yields faster convergence rates.
- Key idea: bias sampling towards Lipschitz constants.
  - "If a gradient can change quickly, sample it more often".
  - "If a gradient can only change slowly, don't sample if often".
- Requires the Lipschitz constant L_i for each example:
  - We use a similar Lipschitz approximation procedure.
  - Adapts to the local distribution of L_i at the solution.

Convergence Result with NUS

- Does SAG converge with NUS?

Convergence Result with NUS

- Does SAG converge with NUS?
- Not known, and seems hard to prove.

Convergence Result with NUS

- Does SAG converge with NUS?
- Not known, and seems hard to prove.
- We showed SAGA converges with NUS [Defazio et al., 2014]:

Proposition: *Let the sequence* $\{w^t\}$ *be defined by*

$$w^{t+1} = w^t - \alpha \left[\frac{1}{p(i)n}(s_j^t - s_j^{t-1}) + \frac{1}{n}\sum_{i=j}^{n} s_j^{t-1} \right],$$

with $\alpha = \frac{np_{min}}{4L+n\mu}$. *Then it holds that*

$$\mathbb{E}[\|w^t - w^*\|^2] \leq \left(1 - \frac{np_{min}\mu}{n\mu + 4L_{max}}\right)^t \left[\|w^0 - w^*\| + T^0\right],$$

Convergence Result with NUS

- Does SAG converge with NUS?
- Not known, and seems hard to prove.
- We showed SAGA converges with NUS [Defazio et al., 2014]:

Proposition: *Let the sequence* $\{w^t\}$ *be defined by*

$$w^{t+1} = w^t - \alpha \left[\frac{1}{p(i)n}(s_j^t - s_j^{t-1}) + \frac{1}{n}\sum_{i=j}^{n} s_j^{t-1} \right],$$

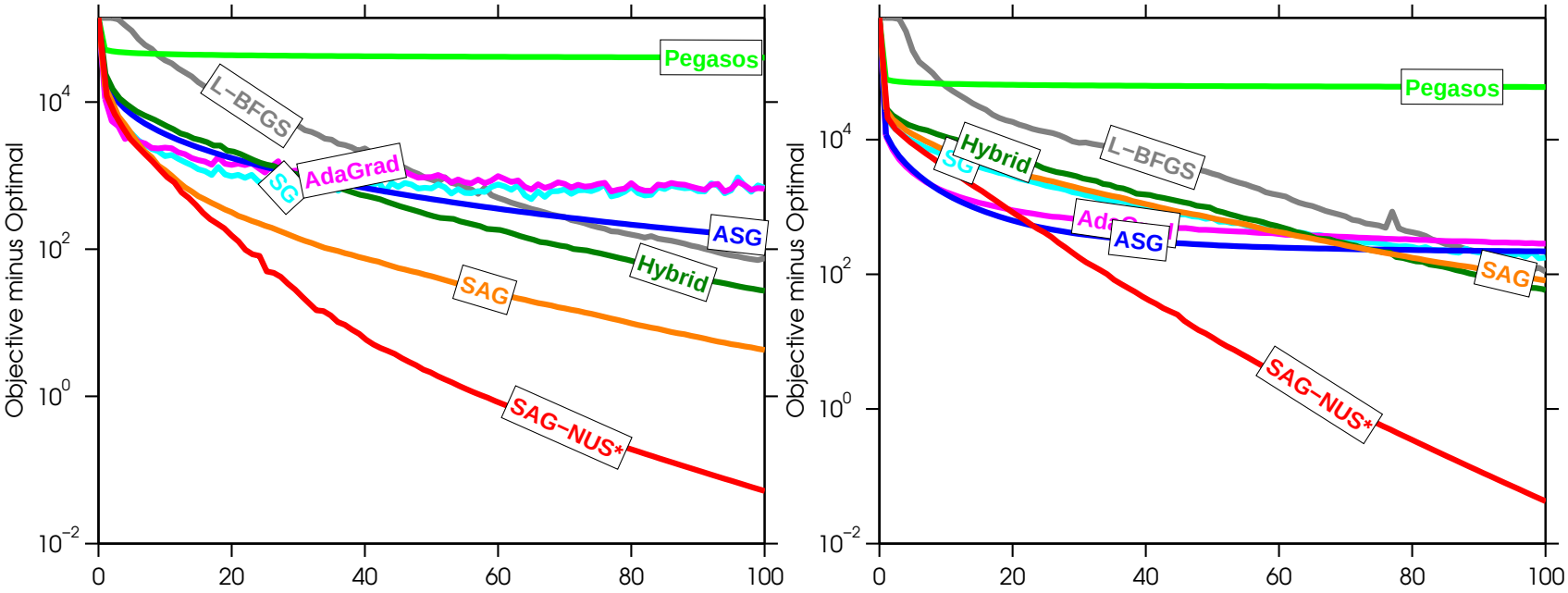
with $\alpha = \frac{np_{min}}{4L+n\mu}$. *Then it holds that*

$$\mathbb{E}[\|w^t - w^*\|^2] \leq \left(1 - \frac{np_{min}\mu}{n\mu + 4L_{max}}\right)^t \left[\|w^0 - w^*\| + T^0\right],$$

- Implies linear convergence rate for any reasonable NUS strategy.

Comparison of SAG-NUS to existing methods

- Comparison of SAG with NUS to existing methods:



(NUS did not improve performance of SG.)

- Consistent and significant improvement.

Discussion

- We explored applying SAG to train CRFs.
- With a few modifications, the memory issue is not an issue.
- Allows adaptive step-size and has a stopping criterion.
- With NUS, substantially improves on state of the art.

Discussion

- We explored applying SAG to train CRFs.
- With a few modifications, the memory issue is not an issue.
- Allows adaptive step-size and has a stopping criterion.
- With NUS, substantially improves on state of the art.

- Could use non-smooth regularizers via proximal/ADMM versions.

  [Mairal, 2013, Defazio et al., 2014, Xiao & Zhang, 2014, Zhong and Kwok, 2013].
- Method should work with approximate inference.
- Method is well-suited to parallel/distributed computation.

Discussion

- We explored applying SAG to train CRFs.
- With a few modifications, the memory issue is not an issue.
- Allows adaptive step-size and has a stopping criterion.
- With NUS, substantially improves on state of the art.

- Could use non-smooth regularizers via proximal/ADMM versions.

  [Mairal, 2013, Defazio et al., 2014, Xiao & Zhang, 2014, Zhong and Kwok, 2013].
- Method should work with approximate inference.
- Method is well-suited to parallel/distributed computation.

- See our poster for a simple analysis showing greedy coordinate descent is faster than random coordinate descent, and how to make it faster (work with Michael Friedlander and Julie Nutini).