Ergodic Subgradient Descent

John Duchi, Alekh Agarwal, Mikael Johansson, Michael Jordan

University of California, Berkeley and Royal Institute of Technology (KTH), Sweden

Allerton Conference, September 2011

Stochastic Gradient Descent

Goal: solve

$$\text{minimize } f(x) \quad \text{subject to } x \in \mathcal{X}.$$

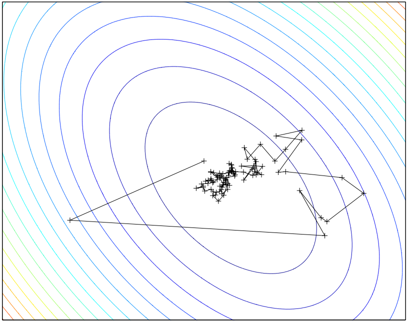
Repeat: At iteration t

- Receive *stochastic gradient* $g(t)$:

$$\mathbb{E}[g(t) \mid g(1), \ldots, g(t-1)] = \nabla f(x(t))$$

- Update

$$x(t+1) = x(t) - \alpha(t) g(t)$$



Stochastic Gradient Descent

Goal: solve

$$\text{minimize } f(x) \quad \text{subject to } x \in \mathcal{X}.$$

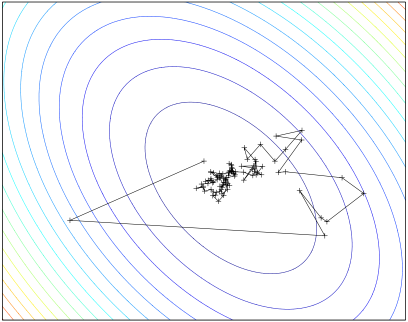
Repeat: At iteration t

- Receive *stochastic gradient* $g(t)$:

$$\mathbb{E}[g(t) \mid g(1), \ldots, g(t-1)] = \underbrace{\nabla f(x(t))}_{\text{unbiased}}$$

- Update

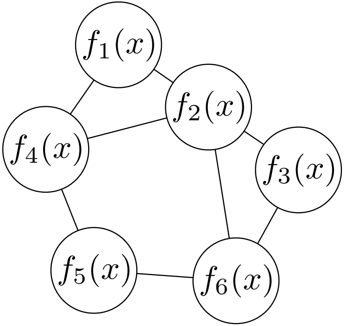
$$x(t+1) = x(t) - \alpha(t) g(t)$$



Peer-to-peer distributed optimization

Setup (Johansson et al. 09): n processors, each possesses function $f_i(x)$. Objective:

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



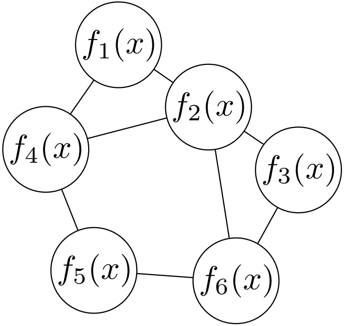
Peer-to-peer distributed optimization

Setup (Johansson et al. 09): n processors, each possesses function $f_i(x)$. Objective:

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

Token walks randomly according to transition matrix P; processor $i(t)$ has token at time t. Update:

$$x(t+1) = \mathrm{Proj}_{\mathcal{X}}\left(x(t) - \alpha \nabla f_{i(t)}(x(t))\right)$$



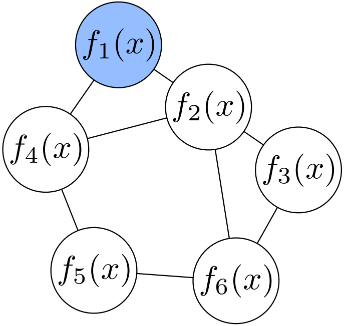
Peer-to-peer distributed optimization

Setup (Johansson et al. 09): n processors, each possesses function $f_i(x)$. Objective:

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

Token walks randomly according to transition matrix P; processor $i(t)$ has token at time t. Update:

$$x(t+1) = \text{Proj}_{\mathcal{X}}\left(x(t) - \alpha \nabla f_{i(t)}(x(t))\right)$$



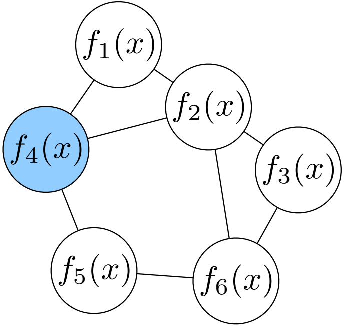
Peer-to-peer distributed optimization

Setup (Johansson et al. 09): n processors, each possesses function $f_i(x)$. Objective:

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

Token walks randomly according to transition matrix P; processor $i(t)$ has token at time t. Update:

$$x(t+1) = \mathrm{Proj}_{\mathcal{X}}\left(x(t) - \alpha \nabla f_{i(t)}(x(t))\right)$$



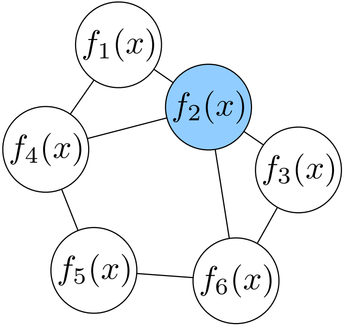
Peer-to-peer distributed optimization

Setup (Johansson et al. 09): n processors, each possesses function $f_i(x)$. Objective:

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

Token walks randomly according to transition matrix P; processor $i(t)$ has token at time t. Update:

$$x(t+1) = \mathrm{Proj}_{\mathcal{X}}\left(x(t) - \alpha \nabla f_{i(t)}(x(t))\right)$$



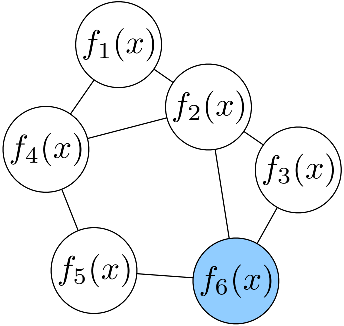
Peer-to-peer distributed optimization

Setup (Johansson et al. 09): n processors, each possesses function $f_i(x)$. Objective:

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

Token walks randomly according to transition matrix P; processor $i(t)$ has token at time t. Update:

$$x(t+1) = \mathrm{Proj}_{\mathcal{X}}\left(x(t) - \alpha \nabla f_{i(t)}(x(t))\right)$$



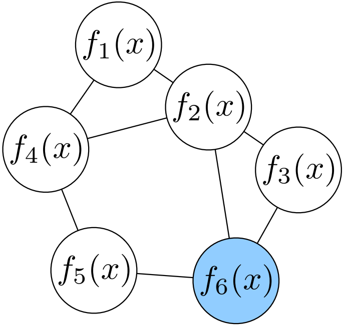
Peer-to-peer distributed optimization

Setup (Johansson et al. 09): n processors, each possesses function $f_i(x)$. Objective:

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

Token walks randomly according to transition matrix P; processor $i(t)$ has token at time t. Update:

$$x(t+1) = \mathrm{Proj}_{\mathcal{X}}\left(x(t) - \alpha \nabla f_{i(t)}(x(t))\right)$$



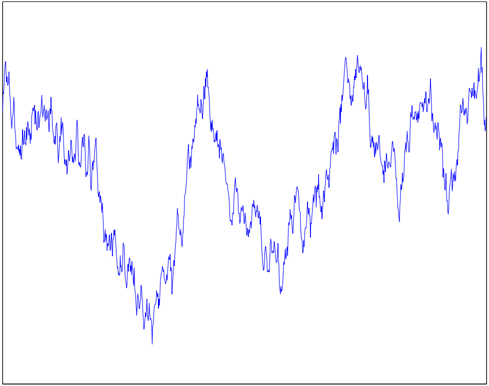
Unbiased gradient estimate: choose $i \in \{1, \ldots, n\}$ uniformly at random,

$$g(t) = \nabla f_i(x(t))$$

Unbiasedness?

Where does the data come from? The noise?

- Financial data
- Autoregressive processes
- Markov chains
- Queuing systems
- Machine learning



Related Work

- Stochastic approximation methods (e.g. Robbins and Monro 1951, Polyak and Juditsky 1992)
- ODE and perturbation methods with dependent noise (e.g. Kushner and Yin 2003)
- Robust approaches and finite sample rates for independent noise settings (Nemirovski and Yudin 1983, Nemirovski et al. 2009)

Related Work

- Stochastic approximation methods (e.g. Robbins and Monro 1951, Polyak and Juditsky 1992)
- ODE and perturbation methods with dependent noise (e.g. Kushner and Yin 2003)
- Robust approaches and finite sample rates for independent noise settings (Nemirovski and Yudin 1983, Nemirovski et al. 2009)

What we do: Finite-time convergence rates of stochastic optimization procedures with dependent noise

Stochastic Optimization

Goal: solve the following problem

$$\min_x \; f(x) = \mathbb{E}_\Pi[F(x;\xi)] = \int_\Xi F(x;\xi)d\Pi(\xi)$$

subject to $x \in \mathcal{X}$

Here ξ come from distribution Π on sample space Ξ.

Stochastic Optimization

Goal: solve the following problem

$$\min_x \; f(x) = \mathbb{E}_\Pi[F(x;\xi)] = \int_\Xi F(x;\xi)d\Pi(\xi)$$

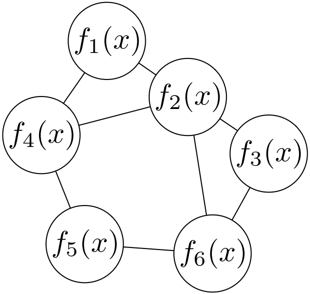
subject to $x \in \mathcal{X}$

Here ξ come from distribution Π on sample space Ξ.

Example: Distributed Optimization

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} F(x;\xi_i) = \frac{1}{n}\sum_{i=1}^{n} f_i(x),$$

where ξ_i is data on machine i



Difficulties

Goal: Solve

$$\min_{x \in \mathcal{X}} \; f(x) = \mathbb{E}_{\Pi}[F(x; \xi)] = \int_{\Xi} F(x; \xi) d\Pi(\xi)$$

Difficulties

Goal: Solve

$$\min_{x \in \mathcal{X}} \; f(x) = \mathbb{E}_{\Pi}[F(x;\xi)] = \int_{\Xi} F(x;\xi) d\Pi(\xi)$$

- Cannot compute expectation in closed form

Difficulties

Goal: Solve

$$\min_{x \in \mathcal{X}} \; f(x) = \mathbb{E}_\Pi[F(x;\xi)] = \int_\Xi F(x;\xi) d\Pi(\xi)$$

- Cannot compute expectation in closed form
- May not know distribution Π

Difficulties

Goal: Solve

$$\min_{x \in \mathcal{X}} \; f(x) = \mathbb{E}_{\Pi}[F(x;\xi)] = \int_{\Xi} F(x;\xi) d\Pi(\xi)$$

- Cannot compute expectation in closed form
- May not know distribution Π
- Might not even be able to *get* samples $\xi \sim \Pi$

Difficulties

Goal: Solve

$$\min_{x \in \mathcal{X}} \; f(x) = \mathbb{E}_{\Pi}[F(x;\xi)] = \int_{\Xi} F(x;\xi) d\Pi(\xi)$$

- Cannot compute expectation in closed form
- May not know distribution Π
- Might not even be able to *get* samples $\xi \sim \Pi$

Solution: stochastic gradient descent, but with samples ξ from a distribution P^t, where

$$P^t \to \Pi$$

Ergodic Gradient Descent

Algorithm: Receive $\xi_t \sim P(\cdot \mid \xi_1, \ldots, \xi_{t-1})$, compute stochastic (sub)gradient:

$$g(t) \in \partial F(x(t); \xi_t),$$

then projected gradient step:

$$x(t+1) = \mathrm{Proj}_{\mathcal{X}} \left[x(t) - \alpha(t) g(t) \right]$$

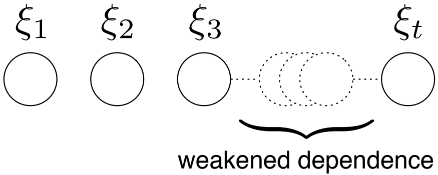
Stochastic Assumption

Ergodicity: The stochastic process $\xi_1, \xi_2, \ldots, \xi_t \sim P$ is sufficiently mixing, i.e.

$$d_{\text{TV}}(P(\xi_t \mid \xi_1, \ldots, \xi_s), \Pi) \to 0$$

as $t - s \uparrow \infty$. Specifically, define *mixing time* τ_{mix} such that

$$t - s \geq \tau_{\text{mix}}(P, \epsilon) \quad \Rightarrow \quad d_{\text{TV}}(P(\xi_t \mid \xi_1, \ldots, \xi_s), \Pi) \leq \epsilon.$$



Main Results:

For algorithm

$$g(t) \in \partial F(x(t); \xi_t), \qquad x(t+1) = \mathrm{Proj}_{\mathcal{X}}\left[x(t) - \alpha(t)g(t)\right]$$

have following guarantees on $\widehat{x}(T) = \frac{1}{T}\sum_{t=1}^{T} x(t)$:

Main Results:

For algorithm

$$g(t) \in \partial F(x(t); \xi_t), \qquad x(t+1) = \mathrm{Proj}_{\mathcal{X}}\left[x(t) - \alpha(t)g(t)\right]$$

have following guarantees on $\widehat{x}(T) = \frac{1}{T}\sum_{t=1}^{T} x(t)$:
Theorem: (Expected convergence) With choice $\alpha(t) \propto 1/\sqrt{t}$,

$$\mathbb{E}[f(\widehat{x}(T))] - f(x^*) \leq C\frac{\sqrt{\tau_{\mathrm{mix}}(P)}}{\sqrt{T}}$$

Main Results:

For algorithm

$$g(t) \in \partial F(x(t); \xi_t), \qquad x(t+1) = \mathrm{Proj}_{\mathcal{X}}\left[x(t) - \alpha(t) g(t)\right]$$

have following guarantees on $\widehat{x}(T) = \frac{1}{T}\sum_{t=1}^{T} x(t)$:
Theorem: (Expected convergence) With choice $\alpha(t) \propto 1/\sqrt{t}$,

$$\mathbb{E}[f(\widehat{x}(T))] - f(x^*) \le C\frac{\sqrt{\tau_{\mathrm{mix}}(P)}}{\sqrt{T}}$$

Theorem: (High-probability convergence) With probability at least $1 - e^{-\kappa}$,

$$f(\widehat{x}(T)) - f(x^*) \le \underbrace{C\frac{\sqrt{\tau_{\mathrm{mix}}(P)}}{\sqrt{T}}}_{\text{Expected rate}} + \underbrace{C\sqrt{\frac{\kappa\tau_{\mathrm{mix}}(P)\log\tau_{\mathrm{mix}}(P)}{T}}}_{\text{Deviation}}$$

Examples:

- Peer-to-peer Distributed Optimization
- Ranking algorithms (optimization on combinatorial spaces)
- Slowly mixing Markov chains
- Any mixing process

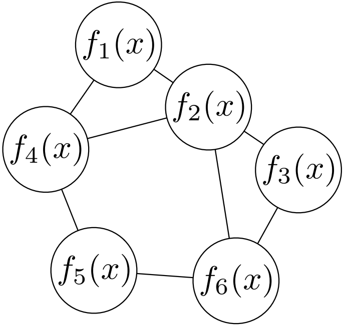
Peer-to-peer distributed optimization

Setup (Johansson et al. 09): n processors, each possesses function $f_i(x)$. Objective:

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

Token walks randomly according to transition matrix P; processor $i(t)$ has token at time t. Update:

$$x(t+1) = \mathrm{Proj}_{\mathcal{X}}\left(x(t) - \alpha \nabla f_{i(t)}(x(t))\right)$$



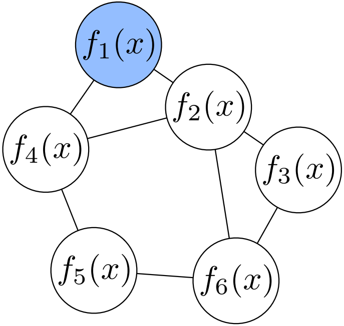
Peer-to-peer distributed optimization

Setup (Johansson et al. 09): n processors, each possesses function $f_i(x)$. Objective:

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

Token walks randomly according to transition matrix P; processor $i(t)$ has token at time t. Update:

$$x(t+1) = \mathrm{Proj}_{\mathcal{X}}\left(x(t) - \alpha \nabla f_{i(t)}(x(t))\right)$$



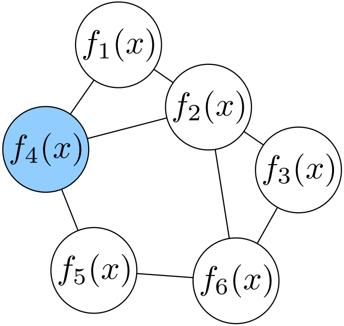
Peer-to-peer distributed optimization

Setup (Johansson et al. 09): n processors, each possesses function $f_i(x)$. Objective:

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

Token walks randomly according to transition matrix P; processor $i(t)$ has token at time t. Update:

$$x(t+1) = \mathrm{Proj}_{\mathcal{X}}\left(x(t) - \alpha \nabla f_{i(t)}(x(t))\right)$$



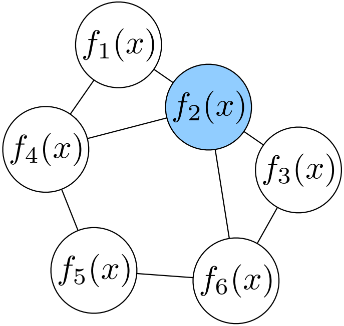
Peer-to-peer distributed optimization

Setup (Johansson et al. 09): n processors, each possesses function $f_i(x)$. Objective:

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

Token walks randomly according to transition matrix P; processor $i(t)$ has token at time t. Update:

$$x(t+1) = \mathrm{Proj}_{\mathcal{X}}\left(x(t) - \alpha \nabla f_{i(t)}(x(t))\right)$$



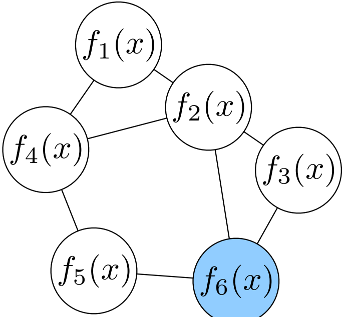
Peer-to-peer distributed optimization

Setup (Johansson et al. 09): n processors, each possesses function $f_i(x)$. Objective:

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

Token walks randomly according to transition matrix P; processor $i(t)$ has token at time t. Update:

$$x(t+1) = \mathrm{Proj}_{\mathcal{X}} \left(x(t) - \alpha \nabla f_{i(t)}(x(t)) \right)$$



Peer-to-peer distributed optimization: Convergence

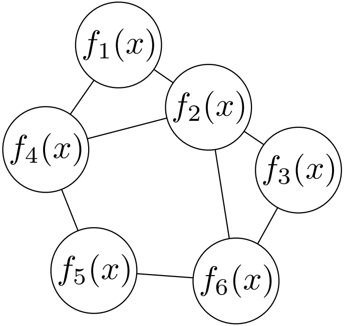
Convergence rate of

$$x(t+1) = \mathrm{Proj}_{\mathcal{X}}\left(x(t) - \alpha(t)\nabla f_{i(t)}(x(t))\right)$$

governed by spectral gap of transition matrix P (here $\rho_2(P)$ is second singular value of P):

$$f(\widehat{x}(T)) - f(x^*) = \widetilde{\mathcal{O}}\left(\underbrace{\sqrt{\frac{\log(Tn)}{1-\rho_2(P)}}}_{\tau_{\mathrm{mix}}} \cdot \frac{1}{\sqrt{T}}\right)$$

in expectation and w.h.p.



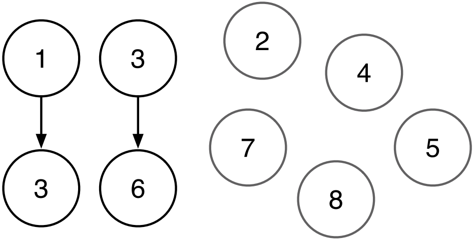
Optimization over combinatorial spaces

General problem: want samples from uniform distribution Π over combinatorial space. Hard to do, so use random walk $P \to \Pi$ (Jerrum & Sinclair)

Optimization over combinatorial spaces

General problem: want samples from uniform distribution Π over combinatorial space. Hard to do, so use random walk $P \to \Pi$ (Jerrum & Sinclair)

Example: Learn a ranking. Receive pairwise user preferences between items, would like to be oblivious to order of remainder. $\mathcal{P}$ is partial order for ranked items, $\{\sigma \in \mathcal{P}\}$ is permutations of $[n]$ consistent with $\mathcal{P}$

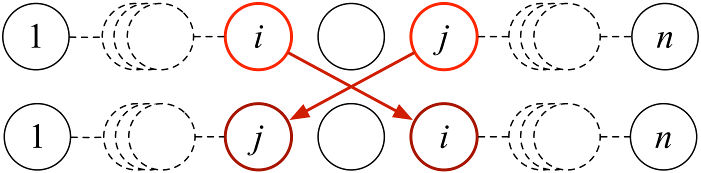


Objective:

$$f(x) := \frac{1}{\operatorname{card}(\sigma \in \mathcal{P})} \sum_{\sigma \in \mathcal{P}} F(x; \sigma)$$

Partial-order Permutation Markov Chain

Markov chain (Karzanov and Khachiyan 91): pick a random pair (i, j), swap if $j \prec i$ is consistent with $\mathcal{P}$



Mixing time (Wilson 04): $\tau_{\mathrm{mix}}(P, \epsilon) \le \frac{4}{\pi^2} n^3 \log \frac{n}{\epsilon}$
Convergence rate:

$$f(\widehat{x}(T)) - f(x^*) = \mathcal{O}\left(\frac{n^{3/2} \sqrt{\log(Tn)}}{\sqrt{T}} \right)$$

Slowly mixing Markov chains

- Might not have such fast mixing rates (i.e. $\log \frac{1}{\epsilon}$)
- Examples:
  - Physical simulations of natural phenomena
  - Autoregressive processes
  - Monte Carlo-sampling based variants of expectation maximization (EM) algorithm

Slowly mixing Markov chains

- Might not have such fast mixing rates (i.e. $\log \frac{1}{\epsilon}$)
- Examples:
  - Physical simulations of natural phenomena
  - Autoregressive processes
  - Monte Carlo-sampling based variants of expectation maximization (EM) algorithm
- Possible mixing rate is *algebraic*:

$$\tau_{\text{mix}}(P, \epsilon) \leq M \epsilon^{-\beta}$$

for a $\beta > 0$.

Slowly mixing Markov chains

- Might not have such fast mixing rates (i.e. $\log \frac{1}{\epsilon}$)
- Examples:
  - Physical simulations of natural phenomena
  - Autoregressive processes
  - Monte Carlo-sampling based variants of expectation maximization (EM) algorithm
- Possible mixing rate is *algebraic*:

$$\tau_{\mathrm{mix}}(P, \epsilon) \leq M \epsilon^{-\beta}$$

  for a $\beta > 0$.
- **Consequence:** expected and high probability convergence

$$f(\widehat{x}(T)) - f(x^*) = \mathcal{O}\left(T^{-\frac{1}{2\beta+2}}\right)$$

Arbitrary mixing process

General convergence guarantee:

$$f(\widehat{x}(T)) - f(x^*) \leq C\frac{1}{\sqrt{T}} + C\inf_{\epsilon>0}\left\{\epsilon + \frac{\tau_{\rm mix}(P,\epsilon)}{\sqrt{T}}\right\}.$$

Consequence: If $\tau_{\rm mix}(P,\epsilon) < \infty$, then

$$f(\widehat{x}(T)) - f(x^*) \to 0 \quad \text{with probability } 1.$$

Conclusions and Discussion

- Finite sample convergence rates for stochastic gradient algorithms with dependent noise
- Convergence rates dependent on mixing time τ_{mix}
- In companion work, we extend these results to *all* stable online learning algorithms (Agarwal and Duchi, 2011)

Conclusions and Discussion

- Finite sample convergence rates for stochastic gradient algorithms with dependent noise
- Convergence rates dependent on mixing time τ_{mix}
- In companion work, we extend these results to *all* stable online learning algorithms (Agarwal and Duchi, 2011)
- Future work: dynamic adaptation for unknown mixing rates
- Weaken uniformity of mixing time assumptions

Thanks!