SVAN 2016 Mini-Course

Stochastic Convex Optimization Methods in Machine Learning

Mark Schmidt

University of British Columbia, May 2016

www.cs.ubc.ca/~schmidtm/SVAN16

Motivation for Parallel and Distributed

- Two recent trends:
  - We aren't making large gains in serial computation speed.
  - Datasets no longer fit on a single machine.

Motivation for Parallel and Distributed

- Two recent trends:
  - We aren't making large gains in serial computation speed.
  - Datasets no longer fit on a single machine.
- Result: we must use parallel and distributed computation.
- Two major issues:
  - Synchronization: we can't wait for the slowest machine.
  - Communication: we can't transfer all information.

Embarassing Parallelism in Machine Learning

- A lot of machine learning problems are embarrassingly parallel:
  - Split task across M machines, solve independently, combine.

Embarassing Parallelism in Machine Learning

- A lot of machine learning problems are embarrassingly parallel:
  - Split task across M machines, solve independently, combine.
- E.g., computing the gradient in deterministic gradient method,

$$\frac{1}{N}\sum_{i=1}^{N}\nabla f_i(x) = \frac{1}{N}\left(\sum_{i=1}^{N/M}\nabla f_i(x) + \sum_{i=(N/M)+1}^{2N/M}\nabla f_i(x) + \dots\right).$$

Embarassing Parallelism in Machine Learning

- A lot of machine learning problems are embarrassingly parallel:
  - Split task across M machines, solve independently, combine.
- E.g., computing the gradient in deterministic gradient method,

$$\frac{1}{N}\sum_{i=1}^{N}\nabla f_i(x) = \frac{1}{N}\left(\sum_{i=1}^{N/M}\nabla f_i(x) + \sum_{i=(N/M)+1}^{2N/M}\nabla f_i(x) + \dots\right).$$

- These allow optimal linear speedups.
  - You should always consider this first!

Asynchronous Computation

- Do we have to wait for the last computer to finish?

Asynchronous Computation

- Do we have to wait for the last computer to finish?
- No!
- Updating asynchronously saves a lot of time.

Asynchronous Computation

- Do we have to wait for the last computer to finish?
- No!
- Updating asynchronously saves a lot of time.
- E.g., stochastic gradient method on shared memory:

$$x^{k+1} = x^k - \alpha \nabla f_{i_k}(x^{k-m}).$$

Asynchronous Computation

- Do we have to wait for the last computer to finish?
- No!
- Updating asynchronously saves a lot of time.
- E.g., stochastic gradient method on shared memory:

$$x^{k+1} = x^k - \alpha \nabla f_{i_k}(x^{k-m}).$$

- You need to decrease step-size in proportion to asynchrony.
- Convergence rate decays elegantly with delay m.[Niu et al., 2011]

Reduced Communication: Parallel Coordinate Descnet

- It may be expensive to communicate parameters x.

Reduced Communication: Parallel Coordinate Descnet

- It may be expensive to communicate parameters x.
- One solution: use parallel coordinate descent:

$$\begin{aligned} x_{j_1} &= x_{j_1} - \alpha_{j_1} \nabla_{j_1} f(x) \\ x_{j_2} &= x_{j_2} - \alpha_{j_2} \nabla_{j_2} f(x) \\ x_{j_3} &= x_{j_3} - \alpha_{j_3} \nabla_{j_3} f(x) \end{aligned}$$

- Only needs to communicate single coordinates.

Reduced Communication: Parallel Coordinate Descnet

- It may be expensive to communicate parameters x.
- One solution: use parallel coordinate descent:

$$x_{j_1} = x_{j_1} - \alpha_{j_1} \nabla_{j_1} f(x)$$
$$x_{j_2} = x_{j_2} - \alpha_{j_2} \nabla_{j_2} f(x)$$
$$x_{j_3} = x_{j_3} - \alpha_{j_3} \nabla_{j_3} f(x)$$

- Only needs to communicate single coordinates.
- Again need to decrease step-size for convergence.
- Speedup is based on density of graph.[Richtarik & Takac, 2013]

Reduced Communication: Decentralized Gradient

- We may need to distribute the data across machines.
- We may not want to update a 'centralized' vector x.

Reduced Communication: Decentralized Gradient

- We may need to distribute the data across machines.
- We may not want to update a 'centralized' vector x.
- One solution: decentralized gradient method:
  - Each processor has its own data samples $f_1, f_2, \ldots f_m$.
  - Each processor has its own parameter vector x_c.
  - Each processor only communicates with a limited number of neighbours $\text{nei}(c)$.

Reduced Communication: Decentralized Gradient

- We may need to distribute the data across machines.
- We may not want to update a 'centralized' vector x.
- One solution: decentralized gradient method:
  - Each processor has its own data samples $f_1, f_2, \ldots f_m$.
  - Each processor has its own parameter vector x_c.
  - Each processor only communicates with a limited number of neighbours $\text{nei}(c)$.

$$x_c = \frac{1}{|\text{nei}(c)|} \sum_{c' \in \text{nei}(c)} x_c - \frac{\alpha_c}{M} \sum_{i=1}^{M} \nabla f_i(x_c).$$

- Gradient descent is special case where all neighbours communicate.
- With modified update, rate decays gracefully as graph becomes sparse.[Shi et al., 2014]
- Can also consider communication failures.[Agarwal & Duchi, 2011]

(pause)

Two Classic Perspectives of Non-Convex Optimization

Two Classic Perspectives of Non-Convex Optimization

- Local non-convex optimization:
  - Apply method with good properties for convex functions.
  - First phase is getting near minimizer.
  - Second phase applies rates from convex optimization.

Two Classic Perspectives of Non-Convex Optimization

- Local non-convex optimization:
  - Apply method with good properties for convex functions.
  - First phase is getting near minimizer.
  - Second phase applies rates from convex optimization.
  - But how long does the first phase take?

Two Classic Perspectives of Non-Convex Optimization

- Local non-convex optimization:
  - Apply method with good properties for convex functions.
  - First phase is getting near minimizer.
  - Second phase applies rates from convex optimization.
  - But how long does the first phase take?
- Global non-convex optimization:
  - Search for global min for general function class.
  - E.g., search over a sucessively-refined grid.
  - Optimal rate for Lipschitz functions is $O(1/\epsilon^{1/D})$.

Two Classic Perspectives of Non-Convex Optimization

- Local non-convex optimization:
  - Apply method with good properties for convex functions.
  - First phase is getting near minimizer.
  - Second phase applies rates from convex optimization.
  - But how long does the first phase take?
- Global non-convex optimization:
  - Search for global min for general function class.
  - E.g., search over a sucessively-refined grid.
  - Optimal rate for Lipschitz functions is $O(1/\epsilon^{1/D})$.
  - Can only solve low-dimensional problems.
- We'll go over recent local, global, and hybrid results..

PL Inequalty: Expanding the Second Phase

- Linear convergence proofs usually assume strong-convexity

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|y - x\|^2.$$

PL Inequalty: Expanding the Second Phase

- Linear convergence proofs usually assume strong-convexity

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|y - x\|^2.$$

- But you can also show linear convergence under many weaker assumptions:
  - Essential strong-convexity, weak strong-convexity, restricted secant inequality, restrictied secant inequality, quadratic growth property, optimal strong-convexity, error bounds.
- In fact, for our proof to work we only required

$$\frac{1}{2}\|\nabla f(x)\|^2 \geq \mu[f(x) - f^*],$$

which we call the Polyak-Łojasiewicz inequality:

PL Inequalty: Expanding the Second Phase

- Linear convergence proofs usually assume strong-convexity

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|y - x\|^2.$$

- But you can also show linear convergence under many weaker assumptions:
  - Essential strong-convexity, weak strong-convexity, restricted secant inequality, restrictied secant inequality, quadratic growth property, optimal strong-convexity, error bounds.
- In fact, for our proof to work we only required

$$\frac{1}{2}\|\nabla f(x)\|^2 \geq \mu[f(x) - f^*],$$

which we call the Polyak-Łojasiewicz inequality:

  - Older than all the above, and also weaker than all the above.

PL Inequalty: Expanding the Second Phase

- Linear convergence proofs usually assume strong-convexity

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|y - x\|^2.$$

- But you can also show linear convergence under many weaker assumptions:
  - Essential strong-convexity, weak strong-convexity, restricted secant inequality, restrictied secant inequality, quadratic growth property, optimal strong-convexity, error bounds.
- In fact, for our proof to work we only required

$$\frac{1}{2}\|\nabla f(x)\|^2 \geq \mu[f(x) - f^*],$$

which we call the Polyak-Łojasiewicz inequality:
  - Older than all the above, and also weaker than all the above.
  - Does not imply solution is unique.
    - Holds for $f(Ax)$ with f strongly-convex even if A is singular.

PL Inequalty: Expanding the Second Phase

- Linear convergence proofs usually assume strong-convexity

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|y - x\|^2.$$

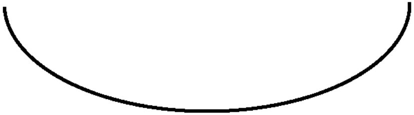
- But you can also show linear convergence under many weaker assumptions:
  - Essential strong-convexity, weak strong-convexity, restricted secant inequality, restrictied secant inequality, quadratic growth property, optimal strong-convexity, error bounds.
- In fact, for our proof to work we only required

$$\frac{1}{2}\|\nabla f(x)\|^2 \geq \mu[f(x) - f^*],$$

which we call the Polyak-Łojasiewicz inequality:

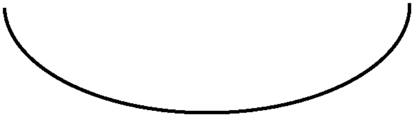
  - Older than all the above, and also weaker than all the above.
  - Does not imply solution is unique.
    - Holds for $f(Ax)$ with f strongly-convex even if A is singular.
  - Does not imply convexity.
  - Also works for coordinate descent, can be generalized to proximal-gradient.

Global Linear Convergence with the PL Inequalty

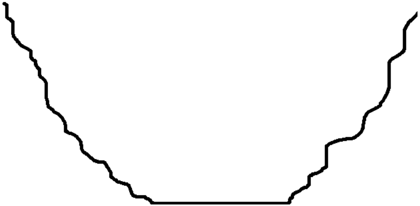


Function satisfying the strong-convexity property:
(unique optimum, convex, growing faster than linear)

Global Linear Convergence with the PL Inequalty



Function satisfying the strong-convexity property:
(unique optimum, convex, growing faster than linear)



Function satisfying the PL inequality:

- Linear convergence rate for this non-convex function.
- Second phase of local solvers is larger than we thought.

General Global Non-Convex Rates?

- For strongly-convex smooth functions, we have

$$\|\nabla f(x^t)\|^2 = O(\rho^t), \quad f(x^t) - f(x^*) = O(\rho^t), \quad \|x_t - x_*\| = O(\rho^t).$$

General Global Non-Convex Rates?

- For strongly-convex smooth functions, we have

$$\|\nabla f(x^t)\|^2 = O(\rho^t), \quad f(x^t) - f(x^*) = O(\rho^t), \quad \|x_t - x_*\| = O(\rho^t).$$

- For convex smooth functions, we have

$$\|\nabla f(x^t)\|^2 = O(1/t), \quad f(x^t) - f(x^*) = O(1/t).$$

General Global Non-Convex Rates?

- For strongly-convex smooth functions, we have

$$\|\nabla f(x^t)\|^2 = O(\rho^t), \quad f(x^t) - f(x^*) = O(\rho^t), \quad \|x_t - x_*\| = O(\rho^t).$$

- For convex smooth functions, we have

$$\|\nabla f(x^t)\|^2 = O(1/t), \quad f(x^t) - f(x^*) = O(1/t).$$

- For non-convex smooth functions, we have

$$\min_k \|\nabla f(x^k)\|^2 = O(1/t).$$

General Global Non-Convex Rates?

- For strongly-convex smooth functions, we have

$$\|\nabla f(x^t)\|^2 = O(\rho^t), \quad f(x^t) - f(x^*) = O(\rho^t), \quad \|x_t - x_*\| = O(\rho^t).$$

- For convex smooth functions, we have

$$\|\nabla f(x^t)\|^2 = O(1/t), \quad f(x^t) - f(x^*) = O(1/t).$$

- For non-convex smooth functions, we have

$$\min_k \|\nabla f(x^k)\|^2 = O(1/t).$$

- You can get this rate for a random iteration of stochastic gradient.
  [Ghadimi & Lan, 2013].

Escaping Saddle Points

- Ghadimi & Lan type of rates could be good or bad news:
  - No dimension dependence (way faster than grid-search).
  - But gives up on optimality (e.g., approximate saddle points).

Escaping Saddle Points

- Ghadimi & Lan type of rates could be good or bad news:
  - No dimension dependence (way faster than grid-search).
  - But gives up on optimality (e.g., approximate saddle points).
- Escaping from saddle points:
  - Classical: trust-region methods allow negative eigenvalues.
  - Modify eigenvalues in Newton's method [Dauphin et al., 2014].
  - Add random noise to stochastic gradient [Ge et al., 2015].

Escaping Saddle Points

- Ghadimi & Lan type of rates could be good or bad news:
  - No dimension dependence (way faster than grid-search).
  - But gives up on optimality (e.g., approximate saddle points).
- Escaping from saddle points:
  - Classical: trust-region methods allow negative eigenvalues.
  - Modify eigenvalues in Newton's method [Dauphin et al., 2014].
  - Add random noise to stochastic gradient [Ge et al., 2015].
  - Cubic regularization of Newton [Nesterov & Polyak, 2006],

$$x^{k+1} = \min_d \left\{ f(x^k) + \langle \nabla f(x^k), d \rangle + \frac{1}{2} d^T \nabla^2 f(x^k) d + \frac{L}{6} \|d\|^3 \right\},$$

  if within ball of saddle point then next step:
    - Moves outside of ball.
    - Has lower objective than saddle-point.

Globally-Optimal Methods for Matrix Problems

Globally-Optimal Methods for Matrix Problems

- Classic: principal component analysis (PCA)

$$\max_{W^T W = I} \|X^T W\|_F^2,$$

and rank-constrained version.
Shamir [2015] gives SAG/SVRG rates for PCA.

Globally-Optimal Methods for Matrix Problems

- Classic: principal component analysis (PCA)

$$\max_{W^TW=I} \|X^TW\|_F^2,$$

  and rank-constrained version.
  Shamir [2015] gives SAG/SVRG rates for PCA.
- Burer & Monteiro [2004] consider SDP re-parameterization

$$\min_{\{X \mid X \succeq 0, \text{rank}(X) \leq k\}} f(X) \Rightarrow \min_V f(VV^T),$$

  and show does not introduce spurious local minimum.

Globally-Optimal Methods for Matrix Problems

- Classic: principal component analysis (PCA)

$$\max_{W^TW=I} \|X^TW\|_F^2,$$

  and rank-constrained version.
  Shamir [2015] gives SAG/SVRG rates for PCA.
- Burer & Monteiro [2004] consider SDP re-parameterization

$$\min_{\{X \mid X \succeq 0, \text{rank}(X) \leq k\}} f(X) \Rightarrow \min_V f(VV^T),$$

  and show does not introduce spurious local minimum.
- De Sa et al. [2015]: For class of non-convex problems of the form

$$\min_Y \mathbb{E}[\|A - VV^T\|_F^2].$$

  random initialization leads to global optimum.

Globally-Optimal Methods for Matrix Problems

- Classic: principal component analysis (PCA)

$$\max_{W^TW=I} \|X^TW\|_F^2,$$

  and rank-constrained version.
  Shamir [2015] gives SAG/SVRG rates for PCA.

- Burer & Monteiro [2004] consider SDP re-parameterization

$$\min_{\{X \mid X\succeq 0, \text{rank}(X)\leq k\}} f(X) \Rightarrow \min_V f(VV^T),$$

  and show does not introduce spurious local minimum.

- De Sa et al. [2015]: For class of non-convex problems of the form

$$\min_Y \mathbb{E}[\|A - VV^T\|_F^2].$$

  random initialization leads to global optimum.

- Under certain assumptions, can solve UV^T dictionary learning and phase retrieval problems [Agarwal et al., 2014, Candes et al., 2015].
- Certain latent variable problems like training HMMs can be solved via SVD and tensor-decomposition methods [Hsu et al., 2012, Anandkumar et al, 2014].

Convex Relaxations/Representations

- Convex relaxations approximate non-convex with convex:
  - Convex relaxations exist for neural nets.
    [Bengio et al., 2005, Aslan et al., 2015].
  - But may solve restricted problem or be a bad approximation.

Convex Relaxations/Representations

- Convex relaxations approximate non-convex with convex:
  - Convex relaxations exist for neural nets.
    [Bengio et al., 2005, Aslan et al., 2015].
  - But may solve restricted problem or be a bad approximation.
- Can solve convex dual:
  - Strong-duality holds for some non-convex problems.
  - Sometimes dual has nicer properties.
  - Efficiently representation/calculation of neural network dual?

Convex Relaxations/Representations

- Convex relaxations approximate non-convex with convex:
  - Convex relaxations exist for neural nets.
    [Bengio et al., 2005, Aslan et al., 2015].
  - But may solve restricted problem or be a bad approximation.
- Can solve convex dual:
  - Strong-duality holds for some non-convex problems.
  - Sometimes dual has nicer properties.
  - Efficiently representation/calculation of neural network dual?
- Exact convex re-formulations of non-convex problems:
  - Laserre [2001].
  - But the size may be enormous.

General Non-Convex Rates

Grid-search is optimal, but can be beaten:

- Convergence rate of Bayesian optimization [Bull, 2011]:
  - Slower than grid-search with low level of smoothness.
  - Faster than grid-search with high level of smoothness:
    - Improves error from $O(1/\epsilon^d)$ to $O(1/\epsilon^{d/\nu})$.

General Non-Convex Rates

Grid-search is optimal, but can be beaten:

- Convergence rate of Bayesian optimization [Bull, 2011]:
  - Slower than grid-search with low level of smoothness.
  - Faster than grid-search with high level of smoothness:
    - Improves error from $O(1/\epsilon^d)$ to $O(1/\epsilon^{d/\nu})$.
- Regret bounds for Bayesian optimization:
  - Exponential scaling with dimensionality [Srinivas et al., 2010].
  - Better under additive assumption [Kandasamy et al., 2015].

General Non-Convex Rates

Grid-search is optimal, but can be beaten:

- Convergence rate of Bayesian optimization [Bull, 2011]:
  - Slower than grid-search with low level of smoothness.
  - Faster than grid-search with high level of smoothness:
    - Improves error from $O(1/\epsilon^d)$ to $O(1/\epsilon^{d/\nu})$.
- Regret bounds for Bayesian optimization:
  - Exponential scaling with dimensionality [Srinivas et al., 2010].
  - Better under additive assumption [Kandasamy et al., 2015].
- Other known faster-than-grid-search rates:
  - Simulated annealing under complicated non-singular assumption [Tikhomirov, 2010].
  - Particle filtering can improve under certain conditions [Crisan & Doucet, 2002].
  - Graduated Non-Convexity for σ-nice functions [Hazan et al., 2014].

Summary

- Parallel and distributed methods will be required in the future.
  - Need asynchronous methods with low communication and fault tolerance.
- We are starting to be able to understand non-convex problems, but there is a lot of work to do.

- Thank you for the invitation and I hope you learned some new things!